NAME

dvisvgm − converts DVI and EPS files to the XML−based SVG format

SYNOPSIS

dvisvgm [*options*] *file* [.dvi]

dvisvgm −E [*options*] *file* [.eps]

DESCRIPTION

The command−line utility **dvisvgm** converts DVI files, as generated by TeX/LaTeX, to the XML−based scalable vector graphics format SVG. It supports the classic DVI format 2 as well as format 3 (created by pTeX in vertical mode), and format 5 which is also known as XDV (created by XeTeX). Besides the basic DVI commands, dvisvgm also evaluates many so−called *specials* which heavily extend the capabilities of the DVI format. For a more detailed overview, see section **Supported Specials** below.

Since the current SVG standard 1.1 doesn't specify multi−page graphics, dvisvgm creates separate SVG files for each DVI page. Because of compatibility reasons, only the first page is converted by default. In order to select a different page or arbitrary page sequences, use option **−p** which is described below.

SVG is a vector−based graphics format and therefore dvisvgm tries to convert the glyph outlines of all used fonts to scalable path descriptions. The fastest way to do that is to extract the path information from font files in PFB, TTF, or OTF format. If dvisvgm is able to find such a file, it extracts all necessary outline information about the glyphs from it.

However, TeX's main source for font descriptions is Metafont, which produces bitmap output (GF files). That's why not all obtainable TeX fonts are available in a scalable format. In these cases, dvisvgm tries to vectorize Metafont's output by tracing the glyph bitmaps. The results are not as perfect as most (manually optimized) PFB or OTF counterparts, but are nonetheless really nice in most cases.

When running dvisvgm without option **−−no−fonts**, *font* elements (<**font**>...</**font**>) are used to embed the font data into the SVG files. Unfortunately, only few SVG renderes support these elements yet. Most web browsers and vector graphics applications don't evaluate them properly so that the text components of the resulting graphics might look strange. In order to create more compatible SVG files, command−line option **−−no−fonts** can be given to replace the font elements by plain graphics paths.

OPTIONS

−a, −−trace−all=[*retrace*]

This option forces dvisvgm to trace not only the actually needed glyphs but all glyphs of all bitmap fonts used in the DVI file. Since the tracing results are stored in the font cache, all following DVI conversions (without option **−−trace−all**) where these fonts are involved, will be much faster. By default, dvisvgm traces only the actually needed glyphs, and adds them to the cache. The boolean option *retrace* determines how to handle glyphs already stored in the cache. By default, these glyphs are skipped. Setting argument *retrace* to *yes* or *true* forces dvisvgm to trace the corresponding bitmaps again.

Note

This option only takes effect if font caching is active. Therefore, **−−trace−all** cannot be

combined with option **−−cache=none**.

−b, −−bbox=*fmt*

Sets the bounding box of the generated graphic to the specified format. The parameter *fmt* takes either one of the format specifiers listed below, or a sequence of four comma− or whitespace−separated length values *x1*, *y1*, *x2* and *y2*. The latter define two diagonal corners of the bounding box. Each length value consists of a floating point number and an optional length unit (pt, bp, cm, mm, in, or pc). If the unit is omitted, TeX points (pt) are assumed.

It's also possible to give only one length value *l*. In this case, the minimal bounding box is computed and enlarged by adding (*−l*,*−l*) to the upper left and (*l*,*l*) to the lower right corner.

Alternatively, the following format specifiers are supported:

International DIN/ISO paper sizes

A*n*, B*n*, C*n*, D*n*, where *n* is a non−negative integer, e.g. A4 or a4 for DIN/ISO A4 format (210mm × 297mm).

North American paper sizes

invoice, executive, legal, letter, ledger

Special bounding box sizes

dvi	page size stored in the DVI file
min	computes the minimal/tightest bounding box
none	no bounding box is assigned

Page orientation

The default page orientation for DIN/ISO and American paper sizes is *portrait*, i.e. *width* < *height*. Appending **−landscape** or simply **−l** to the format string switches to *landscape* mode (*width* > *height*). For symmetry reasons you can also explicitly add **−portrait** or **−p** to indicate the default portrait format. Note that these suffixes are part of the size string and not separate options. Thus, they must directly follow the size specifier without additional blanks. Furthermore, the orientation suffixes can't be used with **dvi**, **min**, and **none**.

Note

Option **−b, −−bbox** only affects the bounding box and does not transform the page content. Hence, if you choose a landscape format, the page won't be rotated.

−C, −−cache[=*dir*]

To speed up the conversion process of bitmap fonts, dvisvgm saves intermediate conversion information in cache files. By default, these files are stored in $HOME/.dvisvgm/cache. If you prefer a different location, use option **−−cache** to overwrite the default. Furthermore, it is also possible to disable the font caching mechanism completely with option **−−cache=none**. If argument *dir* is omitted, dvisvgm prints the path of the default cache directory together with further information about the stored fonts. Additionally, outdated and corrupted cache files are removed.

−j, −−clipjoin

This option tells dvisvgm to compute the intersection of clipping paths itself if necessary,

and not to delegate this task to the SVG renderer. The resulting SVG files are more portable because some SVG renderers don't support intersections of clipping paths which are defined by *clipPath* elements that contain a *clip−path* attribute.

−−color
Enables colorization of messages printed during the conversion process. The colors can be customized via the environment variable **DVISVGM_COLORS**. See the ENVIRONMENT section below for further information.

−E, −−eps
If this option is given, dvisvgm does not expect a DVI but an EPS input file, and tries to convert it to SVG. In order to do so, a single *psfile* special command is created and forwarded to the PostScript special handler. This option is only available if dvisvgm was built with PostScript support enabled, and requires Ghostscript to be available. See option **−−libgs** for further information.

−e, −−exact
If this option is given, dvisvgm computes the precise bounding box of each character. By default, the values stored in a font's TFM file are used to determine a glyph's extent. As these values are intended to implement optimal character placements and are not designed to represent the exact dimensions, they don't necessarily correspond with the bounds of the visual glyphs. Thus, width and/or height of some glyphs may be larger (or smaller) than the respective TFM values. As a result, this can lead to clipped characters at the bounds of the SVG graphics. With option **−−exact** given, dvisvgm analyzes the actual shape of each character and derives a usually tight bounding box.

−m, −−fontmap=*filenames*
Loads and evaluates a single or multiple font map files. These files are required to resolve font file names and encodings. dvisvgm does not provide its own map files but tries to read available ones coming with dvips or dvipdfm. If option **−−fontmap** is omitted, dvisvgm looks for the default map files *ps2pk.map*, *dvipdfm.map*, and *psfonts.map* (in this order). Otherwise, the listed files are used. Multiple filenames must be separated by commas without leading and/or trailing whitespace. The map files are evaluated in the given order. By default, redefined mappings do not replace previous ones. However, each filename can be preceded by an optional mode specifier (+, −, or =) to change this behavior:

+mapfile
Only those entries in the given map file that don't redefine a font mapping are applied. That's also the default mode if no mode specifier is given.

−mapfile
Ensures that none of the font mappings defined in the given map file are used, i.e. previously defined mappings for the specified fonts are removed.

=mapfile
All mappings defined in the map file are applied. Previously defined settings for the same font are replaced.

If the first filename in the filename sequence is preceded by a mode specifier, dvisvgm loads the default font map (see above) and applies the other map files afterwards. Otherwise, none of default map files will be loaded automatically.

Examples: **−−fontmap=myfile1.map,+myfile2.map** loads *myfile1.map* followed by *myfile2.map* where all redefinitions of **myfile2.map** are ignored. **−−fontmap==myfile1.map,−myfile2.map** loads the default map file followed by *myfile1.map* and *myfile2.map* where all redefinitions of *myfile1.map* replace previous entries. Afterwards, all definitions for the fonts given in *myfile2.map* are removed from the font map tree.

For further information about the map file formats and the mode specifiers, see the manuals of dvips and dvipdfm.

−−grad−overlap

Tells dvisvgm to create overlapping grid segments when approximating color gradient fills (also see option **−−grad−segments** below). By default, adjacent segments don't overlap but only touch each other like separate tiles. Unfortunately, this alignment can lead to visible gaps between the segments because the background influences the color at the boundary of the segments if the SVG renderer uses anti−aliasing to create smooth contours. One way to avoid this and to create seamlessly touching color regions is to enlarge the segments so that they extent into the area of their right and bottom neighbors. Since the latter are drawn on top of the overlapping parts, the visible size of all segments keeps unchanged. Just the former gaps disappear as the background is now completely covered by the correct colors. Currently, dvisvgm computes the overlapping segments separately for each patch of the mesh (a patch mesh may consist of multiple patches of the same type). Therefore, there still might be visible gaps at the seam of two adjacent patches.

−−grad−segments=*number*

Determines the maximal number of segments per column and row used to approximate gradient color fills. Since SVG 1.1 only supports a small subset of the shading algorithms available in PostScript, dvisvgm approximates some of them by subdividing the area to be filled into smaller, monochromatic segments. Each of these segments gets the average color of the region it covers. Thus, increasing the number of segments leads to smaller monochromatic areas and therefore results in a better approximation of the actual color gradient. As a drawback, more segments imply bigger SVG files because every segment is represented by a separate path element.

Currently, dvisvgm supports free− and lattice−form triangular patch meshes as well as Coons and tensor−product patch meshes. They are approximated by subdividing the area of each patch into a $n \times n$ grid of smaller segments. The maximal number of segments per column and row can be changed with option **−−grad−segments**.

−−grad−simplify=*delta*

If the size of the segments created to approximate gradient color fills falls below the given delta value, dvisvgm reduces their level of detail. For example, Bézier curves are replaced by straight lines, and triangular segments are combined to tetragons. For a small delta these simplifications are usually not noticeable but reduce the size of the generated SVG files.

−h, −−help[=*mode*]

Prints a short summary of all available command−line options. The optional *mode* parameter is an integer value between 0 and 2. It selects the display variant of the help text. Mode 0 lists all options divided into categories with section headers. This is also the default if dvisvgm is called without parameters. Mode 1 lists all options ordered by the short option names, while mode 2 sorts the lines by the long option names.

−−keep
Disables the removal of temporary files as created by Metafont (usually .gf, .tfm, and .log files).

−−libgs=*filename*
This option is only available if the Ghostscript library is not directly linked to dvisvgm and if PostScript support was not completely disabled during compilation. In this case, dvisvgm tries to load the shared GS library dynamically during runtime. By default, it expects the library's name to be libgs.so.X (on Unix−like systems, where X is the ABI version of the library) or gsdll32.dll/gsdll64.dll (Windows). Option **−−libgs** can be used to give a different name. Alternatively, it's also possible to set the GS library name by the environment variable **LIBGS**. The latter has less precedence than the command−line option, i.e. dvisvgm ignores variable **LIBGS** if **−−libgs** is given.

−L, −−linkmark=*style*
Selects the method how to mark hyperlinked areas. The *style* argument can take one of the values *none*, *box*, and *line*, where *box* is the default, i.e. a rectangle is drawn around the linked region if option **−−linkmark** is omitted. Style argument *line* just draws the lower edge of the bounding rectangle, and *none* tells dvisvgm not to add any visible objects to hyperlinks. The lines and boxes get the current text color selected. In order to apply a different, constant color, a colon followed by a color specifier can be appended to the style string. A *color specifier* is either a hexadecimal RGB value of the form *#RRGGBB*, or a dvips color name (http://en.wikibooks.org/wiki/LaTeX/Colors#The_68_standard_colors_known_to_dvips).

Moreover, argument *style* can take a single color specifier to highlight the linked region by a frameless box filled with that color. An optional second color specifier separated by colon selects the frame color.

Examples: **box:red** or **box:#ff0000** draws red boxes around the linked areas. **yellow:blue** creates yellow filled rectangles with blue frames.

−l, −−list−specials
Prints a list of registered special handlers and exits. Each handler processes a set of special statements belonging to the same category. In most cases, the categories are identified by the prefix of the special statements. It's usually a leading word separated from the rest of the statement by a colon or a blank, e.g. *color* or *ps*.

−M, −−mag=*factor*
Sets the magnification factor applied in conjunction with Metafont calls prior tracing the glyphs. The larger this value, the better the tracing results. Nevertheless, large magnification values can cause Metafont arithmetic errors due to number overflows. So, use this option with care. The default setting usually produces nice results.

−−no−merge
Puts every single character in a separate *text* element with corresponding *x* and *y* attributes. By default, new *text* or *tspan* elements are only created if a string starts at a location that differs from the regular position defined by the characters' advance values.

−−no−mktexmf
Suppresses the generation of missing font files. If dvisvgm can't find a font file through the kpathsea lookup mechanism, it calls the external tools mktextfm or mktexmf by. This option

disables these calls.

−n, −−no−fonts[=*variant*]
If this option is given, dvisvgm doesn't create SVG *font* elements but uses *paths* instead. The resulting SVG files tend to be larger but they are concurrently more compatible with most applications that don't support SVG fonts yet. The optional argument *variant* selects the method how to substitute fonts by paths. Variant 0 creates *path* and *use* elements. Variant 1 creates *path* elements only. Option **−−no−fonts** implies **−−no−styles**.

−c, −−scale=*sx*[,*sy*]
Scales the page content horizontally by *sx* and vertically by *sy*. This option is equivalent to **−TS***sx*,*sy*.

−S, −−no−specials[=*names*]
Disable processing of special commands embedded in the DVI file. If no further parameter is given, all specials are ignored. To selectively disable sets of specials, an optional comma−separated list of names can be appended to this option. A *name* is the unique identifier referencing the intended special handler. Option **−−list−specials** lists all currently available handlers and their names. All unsupported special statements are silently ignored.

−−no−styles
By default, dvisvgm creates CSS styles and class attributes to reference fonts because it's more compact than repeatedly set the complete font information in each text element. However, if you prefer direct font references, the default behavior can be disabled with option **−−no−styles**.

−o, −−output=*pattern*
Sets the name pattern of the output file. Parameter *pattern* is a string that may contain the variables **%f**, **%p**, and **%P**. **%f** expands to the base name of the DVI file, i.e. the filename without suffix, **%p** is the current page number, and **%P** the total number of pages in the DVI file. An optional number (0−9) given after the percent sign specifies the minimal number of digits to be written. If a particular value is shorter, the number is padded with leading zeros. Example: **%3p** enforces 3 digits for the current page number (001, 002, etc.). Without an explicit width specifier, **%p** gets the same number of digits as **%P**.

If you need more control over the numbering, you can use arithmetic expressions as part of a pattern. The syntax is **%(expr)** where *expr* may contain additions, subtractions, multiplications, and integer divisions with common precedence. The variables **p** and **P** contain the current page number and the total number of pages, respectively. For example, **−−output="%f−%(p−1)"** creates filenames where the numbering starts with 0 rather than 1.

The default pattern is **%f−%p.svg** if the DVI file consists of more than one page, and **%f.svg** otherwise. That means, a DVI file *foo.dvi* is converted to *foo.svg* if *foo.dvi* is a single−page document. Otherwise, multiple SVG files *foo−01.svg*, *foo−02.svg*, etc. are produced. In Windows environments, the percent sign indicates dereferenced environment variables, and must therefore be protected by a second percent sign, e.g. **−−output=%%f−%%p**.

−p, −−page=*ranges*
This option sets the pages to be processed. Parameter *ranges* consists of a comma−separated list of single page numbers and/or page ranges. A page range is a pair of numbers separated

by a hyphen, e.g. 5−12. Thus, a page sequence might look like this: 2−4,6,9−12,15. It doesn't matter if a page is given more than once or if page ranges overlap. dvisvgm always extracts the page numbers in ascending order and converts them only once. In order to stay compatible with previous versions, the default page sequence is 1. dvisvgm therefore converts only the first page and not the whole document in case option **−−page** is omitted. Usually, page ranges consist of two numbers denoting the first and last page to be converted. If the conversion is to be started at page 1, or if it should continue up to the last DVI page, the first or second range number can be omitted, respectively. Example: **−−page=−10** converts all pages up to page 10, **−−page=10−** converts all pages starting with page 10. Please consider that the page values don't refer to the page numbers printed on the page. Instead, the physical page count is expected, where the first page always gets number 1.

−d, −−precision=*digits*
Specifies the maximal number of decimal places applied to floating−point attribute values. All attribute values written to the generated SVG file(s) are rounded accordingly. The parameter *digits* allows integer values from 0 to 6, where 0 enables the automatic selection of significant decimal places. This is also the default value if dvisvgm is called without option **−−precision**.

−P, −−progress[=*delay*]
Enables a simple progress indicator shown when time−consuming operations like PostScript specials are processed. The indicator doesn't appear before the given delay (in seconds) has elapsed. The default delay value is 0.5 seconds.

−r, −−rotate=*angle*
Rotates the page content clockwise by *angle* degrees around the page center. This option is equivalent to **−TR***angle*.

−R, −−relative
SVG allows to define graphics paths by a sequence of absolute and/or relative commands, i.e. each command expects either absolute coordinates or coordinates relative to the current drawing position. By default, dvisvgm creates paths made up of absolute commands. If option **−−relative** is given, relative commands are created instead which slightly reduces the size of the SVG files in most cases.

−s, −−stdout
Don't write the SVG output to a file but redirect it to **stdout**.

−T, −−transform=*commands*
Applies a sequence of transformations to the SVG content. Each transformation is described by a *command* beginning with a capital letter followed by a list of comma−separated parameters. Following transformation commands are supported:

T *tx*[,*ty*]
Translates (moves) the page in direction of vector (*tx*,*ty*). If *ty* is omitted, *ty*=0 is assumed. The expected unit length of *tx* and *ty* are TeX points (1pt = 1/72.27in). However, there are several constants defined to simplify the unit conversion (see below).

S *sx*[,*sy*]
Scales the page horizontally by *sx* and vertically by *sy*. If *sy* is omitted, *sy*=*sx* is assumed.

R *angle*[,*x*,*y*]

Rotates the page clockwise by *angle* degrees around point (*x*,*y*). If the optional arguments *x* and *y* are omitted, the page will be rotated around its center depending on the chosen page format. When option **−bnone** is given, the rotation center is origin (0,0).

KX *angle*
Skews the page along the *x*−axis by *angle* degrees. Argument *angle* can take any value except 90+180*k*, where *k* is an integer.

KY *angle*
Skews the page along the *y*−axis by *angle* degrees. Argument *angle* can take any value except 90+180*k*, where *k* is an integer.

FH [*y*]
Mirrors (flips) the page at the horizontal line through point (0,*y*). Omitting the optional argument leads to *y*=*h*/2, where *h* denotes the page height (see *pre−defined constants* below).

FV [*x*]
Mirrors (flips) the page at the vertical line through point (*x*,0). Omitting the optional argument leads to *x*=*w*/2, where *w* denotes the page width (see *pre−defined constants* below).

M *m1*,...,*m6*
Applies a transformation described by the 3×3 matrix ((*m1*,*m2*,*m3*),(*m4*,*m5*,*m6*),(0,0,1)), where the inner triples denote the rows.

Note

All transformation commands of option **−T, −−transform** are applied in the order of their appearance. Multiple commands can optionally be separated by spaces. In this case the whole transformation string has to be enclosed in double quotes. All parameters are expressions of floating point type. You can either give plain numbers or arithmetic terms combined by the operators **+** (addition), **−** (subtraction), ***** (multiplication), **/** (division) or **%** (modulo) with common associativity and precedence rules. Parentheses may be used as well.

Additionally, some pre−defined constants are provided:

ux	horizontal position of upper left page corner in TeX point units
uy	vertical position of upper left page corner in TeX point units
h	page height in TeX point units (0 in case of **−bnone**)
w	page width in TeX point units (0 in case of **−bnone**)

Furthermore, you can use the length constants **pt**, **mm**, **cm** and **in**, e.g. **2cm** or **1.6in**. Thus, option **−TT1in,0R45** moves the page content 1 inch to the right and rotates it by 45 degrees around the page center afterwards.

For single transformations you can also use options **−c**, **−t** and **−r**. Note that the order in which these options are given is not significant, i.e. you can't use them to describe

transformation sequences. They are simply independent shorthand options for common transformations.

−t, −−translate=*tx*[,*ty*]
Translates (moves) the page content in direction of vector (*tx*,*ty*). This option is equivalent to **−TT***tx*,*ty*.

−v, −−verbosity=*level*
Controls the type of messages printed during a dvisvgm run:

0 no message output
1 error messages only
2 warning messages only
4 informational messages only

Note
By adding these values you can combine the categories. The default level is 7, i.e. all messages are printed.

−V, −−version[=*extended*]
Prints the version of dvisvgm and exits. If the optional argument is set to *yes*, the version numbers of the linked libraries are printed as well.

−z, −−zip[=*level*]
Creates a compressed SVG file with suffix .svgz. The optional argument specifies the compression level. Valid values are in the range of 1 to 9 (default value is 9). Larger values cause better compression results but take more computation time.

Caution
This option cannot be combined with **−s, −−stdout**.

−Z, −−zoom[=*factor*]
Multiplies the *width* and *height* attributes of the SVG root element by argument *factor* while the coordinate system of the graphic is retained. As a result, most SVG viewers zoom the graphics accordingly. If a negative zoom factor is given, the *width* and *height* attributes are omitted.

SUPPORTED SPECIALS

dvisvgm supports several sets of *special commands* that can be used to enrich DVI files with additional features, like color, graphics, and hyperlinks. The evaluation of special commands is delegated to dedicated handlers. Each handler is responsible for all special statements of the same command set, i.e. commands beginning with the same prefix. To get a list of actually provided special handlers, use option **−−list−specials** (see above). This section gives an overview of the special commands currently supported.

bgcolor
Special statement for changing the background/page color. Since SVG 1.1 doesn't support background colors, dvisvgm inserts a rectangle of the chosen color into the generated SVG document. This rectangle always gets the same size as the selected or computed bounding box. This background color command is part of the color special set but is handled separately in order to let the user turn it off. For an overview of the command syntax, see the documentation of dvips, for instance.

color

Statements of this command set provide instructions to change the text/paint color. For an overview of the exact syntax, see the documentation of dvips, for instance.

dvisvgm

dvisvgm offers its own small set of specials. The following list gives a brief overview.

dvisvgm:raw *text*

Adds an arbitrary sequence of characters to the page section of the SVG document. dvisvgm does not perform any validation here, thus the user has to ensure that the resulting SVG is still valid. Parameter *text* may contain the expressions **{?x}**, **{?y}**, and **{?color}** that expand to the current *x* or *y* coordinate and the current color, respectively. Furthermore, **{?nl}** expands to a newline character.

dvisvgm:rawdef *text*

This command is similar to **dvisvgm:raw**, but puts the raw text into the <defs> section of the SVG document currently being generated.

dvisvgm:rawset *name* ... **dvisvgm:endrawset**

This pair of specials marks the begin and end of a definition of a named raw SVG fragment. All **dvisvgm:raw** and **dvisvgm:rawdef** specials enclosed by **dvisvgm:rawset** and **dvisvgm:endrawset** are not evaluated immediately but jointly stored under the given *name* for later use. Once defined, the named fragment can be referenced throughout the DVI file by **dvisvgm:rawput** (see below). The two commands **dvisvgm:rawset** and **dvisvgm:endrawset** must not be nested, i.e. each call of **dvisvgm:rawset** has to be followed by a corresponding call of **dvisvgm:endrawset** before another **dvisvgm:rawset** may occur. Also, the identifier *name* must be unique throughout the DVI file. Using **dvisvgm:rawset** multiple times together with the same *name* leads to a warning message.

dvisvgm:rawput *name*

Inserts raw SVG fragments previously stored under the given *name*. dvisvgm distinguishes between fragments that were specified with **dvisvgm:raw** or **dvisvgm:rawdef**, and handles them differently: It inserts all **dvisvgm:raw** parts every time **dvisvgm:rawput** is called, whereas the **dvisvgm:rawdef** portions go to the <defs> section of the current SVG document only once.

dvisvgm:img *width height file*

Creates an image element at the current graphic position referencing the given file. JPEG, PNG, and SVG images can be used here. However, dvisvgm does not check the file format or the file name suffix. The lengths *width* and *height* must be given as plain floating point numbers in TeX point units (1in = 72.27pt).

dvisvgm:bbox n[ew] *name*

Defines or resets a local bounding box called *name*. The name may consist of letters and digits. While processing a DVI page, dvisvgm continuously updates the (global) bounding box of the current page in order to determine the minimal rectangle containing all visible page components (characters, images, drawing elements etc.) Additionally to the global bounding box, the user can request an arbitrary number of named local bounding boxes. Once defined, these boxes are updated together with the global bounding box starting with the first character that follows the definition. Thus, the local boxes can be used to compute the extent of parts of the page. This is useful for

scenarios where the generated SVG file is post−processed. In conjunction with special dvisvgm:raw, the macro **{?bbox** *name***}** expands to the four values *x*, *y*, *w*, and *h* (separated by spaces) specifying the coordinates of the upper left corner, width, and height of the local box *name*. If box *name* wasn't previously defined, all four values equal zero.

dvisvgm:bbox *width height* [*depth*]
Updates the bounding box of the current page by embedding a virtual rectangle (*x*, *y*, *width*, *height*) where the lower left corner is located at the current DVI drawing position (*x*,*y*). If the optional parameter *depth* is specified, dvisvgm embeds a second rectangle (*x*, *y*, *width*, −*depth*). The lengths *width*, *height* and *depth* must be given as plain floating point numbers in TeX point units (1in = 72.27pt). Depending on size and position of the virtual rectangle, this command either enlarges the overall bounding box or leaves it as is. It's not possible to reduce its extent. This special should be used in conjunction with **dvisvgm:raw** in order to update the viewport of the page properly.

dvisvgm:bbox a[bs] *x1 y1 x2 y2*
This variant of the bbox special updates the bounding box by embedding a virtual rectangle (*x1*,*y1*,*x2*,*y2*). The points (*x1*,*y1*) and (*x2*,*y2*) denote two diagonal corners of the rectangle given in TeX point units.

dvisvgm:bbox f[ix] *x1 y1 x2 y2*
This variant of the bbox special assigns an absolute (final) bounding box to the resulting SVG. After executing this command, dvisvgm doesn't further alter the bounding box coordinates, except this special is called again later. The points (*x1*,*y1*) and (*x2*,*y2*) denote two diagonal corners of the rectangle given in TeX point units.

The following TeX snippet adds two raw SVG elements to the output and updates the bounding box accordingly:

```
\special{dvisvgm:raw <circle cx='{?x}' cy='{?y}' r='10' stroke='black' fill='red'/>}
\special{dvisvgm:bbox 20 10 10}

\special{dvisvgm:raw <path d='M50 200 L10 250 H100 Z' stroke='black' fill='blue'/>}
\special{dvisvgm:bbox abs 10 200 100 250}
```

em
These specials were introduced with the emTeX distribution by Eberhard Mattes. They provide line drawing statements, instructions for embedding MSP, PCX, and BMP image files, as well as two PCL commands. dvisvgm supports only the line drawing statements and ignores all other em specials silently. A description of the command syntax can be found in the DVI driver documentation coming with emTeX (see CTAN).

html
The hyperref specification defines several variants on how to mark hyperlinked areas in a DVI file. dvisvgm supports the plain HyperTeX special constructs as created with hyperref package option *hypertex*. By default, all linked areas of the document are marked by a rectangle. Option **−−linkmark** allows to change this behavior. See above for further details. Information on syntax and semantics of the HyperTeX specials can be found in the hyperref manual.

pdf

pdfTeX and dvipdfmx introduced several special commands related to the generation of PDF files. Currently, only two of them, *pdf:mapfile* and *pdf:mapline* are supported by dvisvgm. These specials allow modifying the font map tree during the processing of DVI files. They are used by CTeX, for example. dvisvgm supports both, the dvips and dvipdfm font map format. For further information on the command syntax and semantics, see the documentation of **\pdfmapfile** in the pdfTeX user manual.

ps

The famous DVI driver dvips introduced its own set of specials in order to embed PostScript code into DVI files, which greatly improves the capabilities of DVI documents. One aim of dvisvgm is to completely evaluate all PostScript snippets and to convert as many of them as possible to SVG. In contrast to dvips, dvisvgm uses floating point arithmetics to compute the precise position of each graphic element, i.e. it doesn't round the coordinates. Therefore, the relative locations of the graphic elements may slightly differ from those computed by dvips.

Since PostScript is a rather complex language, dvisvgm does not try to implement its own PostScript interpreter but relies on Ghostscript (http://ghostscript.com) instead. If the Ghostscript library was not linked to the dvisvgm binary, it is looked up and loaded dynamically during runtime. In this case, dvisvgm looks for *libgs.so.X* on Unix−like systems (supported ABI versions: 7,8,9), and for *gsdll32.dll* or *gsdll64.dll* on Windows. You can override the default file names with environment variable **LIBGS** or the command−line option **−−libgs**. The library must be reachable through the ld search path (*nix) or the PATH environment variable (Windows). Alternatively, the absolute file path can be specified. If the library cannot be found, dvisvgm disables the processing of PostScript specials and prints a warning message. Use option **−−list−specials** to check whether PostScript support is available, i.e. entry *ps* is present.

The PostScript handler also recognizes and evaluates bounding box data generated by the *preview* package with option *tightpage*. If the data is present in a DVI file, dvisvgm adapts the bounding box of the generated SVG file accordingly, and prints a message showing the width, height, and depth of the box in TeX point units. Especially, the depth value can be used to vertically align the SVG graphics with the baseline of surrounding text in HTML or XSL−FO documents, for example.

tpic

The TPIC special set defines instructions for drawing simple geometric objects. Some LaTeX packages, like eepic and tplot, use these specials to describe graphics.

EXAMPLES

dvisvgm file

Converts the first page of *file.dvi* to *file.svg*.

dvisvgm −z file

Converts the first page of *file.dvi* to *file.svgz* with default compression level 9.

dvisvgm −p5 −z3 −ba4−l −onewfile file

Converts the fifth page of *file.dvi* to *newfile.svgz* with compression level 3. The bounding box is set to DIN/ISO A4 in landscape format.

dvisvgm −−transform="R20,w/3,2h/5 T1cm,1cm S2,3" file

Converts the first page of *file.dvi* to *file.svg* where three transformations are applied.

ENVIRONMENT

dvisvgm uses the **kpathsea** library for locating the files that it opens. Hence, the environment variables described in the library's documentation influence the converter.

If dvisvgm was linked without the Ghostscript library, and if PostScript support has not been disabled, the shared Ghostscript library is looked up during runtime via dlopen(). The environment variable **LIBGS** can be used to specify path and file name of the library.

The pre−compiled Windows versions of dvisvgm require a working installation of MiKTeX 2.9 or above. dvisvgm does not work together with the portable edition of MiKTeX because it relies on MiKTeX's COM interface only accessible in a local installation. To enable the evaluation of PostScript specials, the original Ghostscript DLL *gsdll32.dll* must be present and reachable through the search path. 64−bit Windows builds require the 64−bit Ghostscript DLL *gsdll64.dll*. Both DLLs come with the corresponding Ghostscript installers available from www.ghostscript.com.

The environment variable **DVISVGM_COLORS** specifies the colors used to highlight various parts of dvisvgm's message output. It is only evaluated if option **−−color** is given. The value of **DVISVGM_COLORS** is a list of colon−separated entries of the form *gg=BF*, where *gg* denotes one of the color group indicators listed below, and *BF* are two hexadecimal digits specifying the background (first digit) and foreground/text color (second digit). The color values are defined as follows: 0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=gray, 8=bright red, 9=bright green, A=bright yellow, B=bright blue, C=bright magenta, D=bright cyan, E=bright gray, F=white. Depending on the terminal, the colors may differ. Rather than changing both the text and background color, it's also possible to change only one of them: An asterisk (*) in place of a hexadecimal digit indicates the default text or background color of the terminal.

All malformed entries in the list are silently ignored.

er error messages

wn warning messages

pn messages about page numbers

ps page size messages

fw information about the files written

sm state messages

tr messages of the glyph tracer

pi progress indicator

Example: er=01:pi=*5 sets the colors of error messages (**er**) to red (**1**) on black (**0**), and those of progress indicators (**pi**) to cyan (**5**) on default background (*).

FILES

The location of the following files is determined by the kpathsea library. To check the actual kpathsea configuration you can use the **kpsewhich** utility.

***.enc** Font encoding files

***.fgd** Font glyph data files (cache files created by dvisvgm)

***.map** Font map files

***.mf** Metafont input files

***.pfb** PostScript Type 1 font files

***.pro** PostScript header/prologue files

***.tfm** TeX font metric files

***.ttf** TrueType font files

***.vf** Virtual font files

SEE ALSO

tex(1), mf(1), mktexmf(1), grodvi(1), potrace(1), and the **kpathsea library** info documentation.

RESOURCES

Project home page

http://dvisvgm.sourceforge.net

SourceForge project site

http://sourceforge.net/projects/dvisvgm

BUGS

Please report bugs using the bug tracker at Launchpad (https://launchpad.net/dvisvgm) or GitHub (https://github.com/mgieseki/dvisvgm).

AUTHOR

Written by Martin Gieseking <**martin.gieseking@uos.de**>

COPYING

Copyright © 2005−2015 Martin Gieseking. Free use of this software is granted under the terms of the GNU General Public License (GPL) version 3 or, (at your option) any later version.