CLX-TRUETYPE

CLX TrueType Font Renderer

Table of Contents

Introduction

CLX-TRUETYPE renders TrueType fonts over X11 drawable (window or pixmap) using CLX, XRender, ZPB-TTF, CL-VECTORS.

CLX-TRUETYPE was originally written for mcclim font rendering by Gilbert Baumann and Andy Hefner.

CLX-TRUETYPE is maintained in Git:

```
git clone git://github.com/filonenko-mikhail/clx-truetype
```

will get you a local copy.

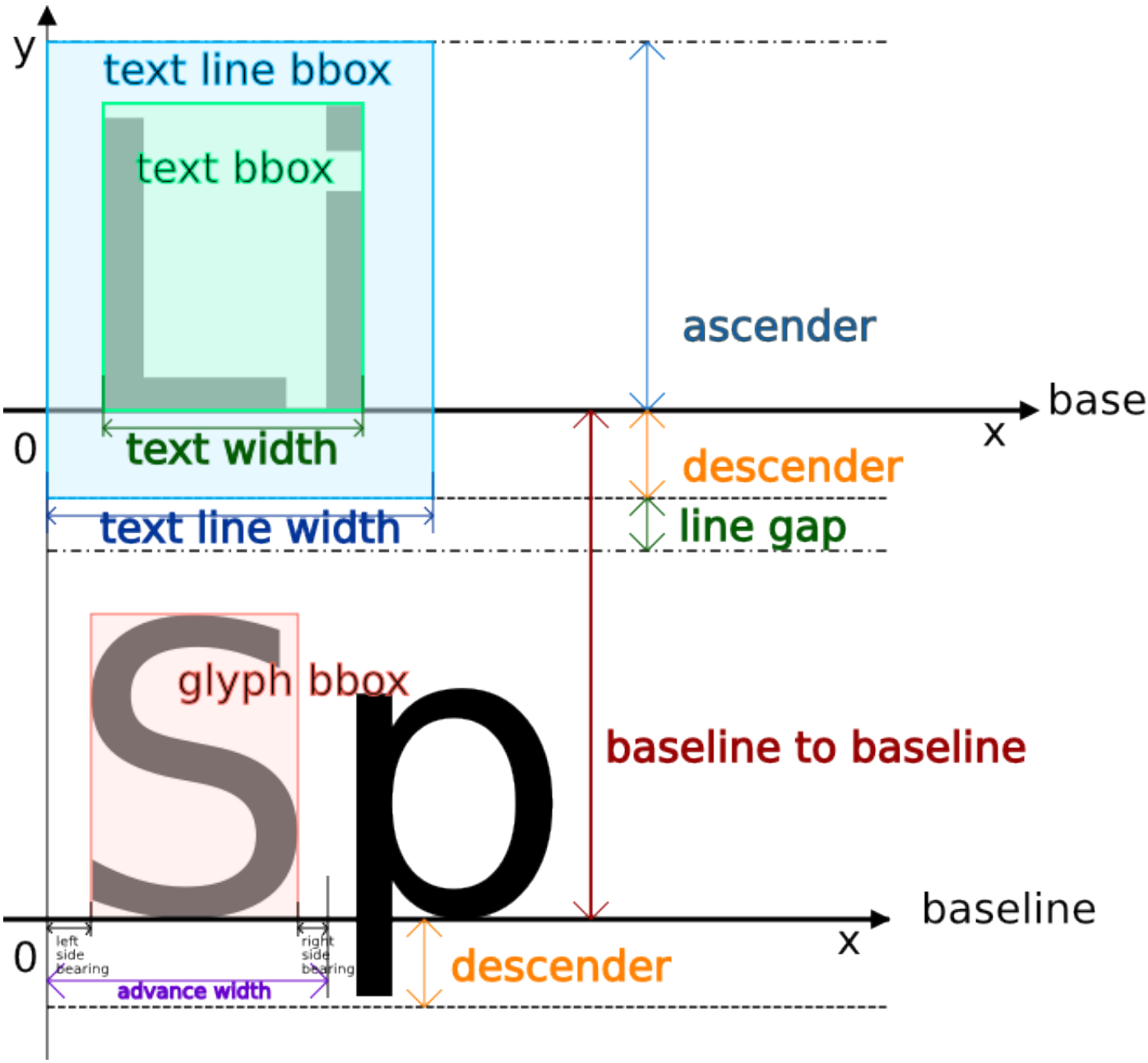
http://github.com/filonenko-mikhail/clx-truetype/

is the GitHub project page, where the issue tracker is located.

1 Overview

CLX-TRUETYPE is library for text rendering over X11 drawable using CLX, XRender, ZPB-TTF, CL-VECTORS.

TrueType font metrics



- TrueType hints are not supported.
- RGB antialiasing is not supported.
- Text rendering do not use XRender glyph sets.

2 Examples

Drawing text is quite simple.

First and only one time step is loading font cache using [Function cache-fonts], page 6. If you add font to your system, you should call it again.

```
(cache-fonts)
```

Make instance of font:

```
(font (make-instance 'font :family "Times New Roman" :subfamily "Bold Italic"
                           :size 12 :antialiased t))
```

Draw it using [Function draw-text], page 8 or [Function draw-text-line], page 8 functions:

```
(draw-text window grackon font "The quick brown fox jumps over the lazy dog." 100 100)
```

Move <<cursor>> using [Function baseline-to-baseline], page 7 distance.

Here it is complete example. Just insert it into repl, and evaluate (show-window).

```
(defpackage #:clx-truetype-test
  (:nicknames :xft-test)
  (:use #:cl #:xft)
  (:export show-window))

(in-package :clx-truetype-test)

(defvar *display* (xlib:open-default-display))

(defvar *screen* (xlib:display-default-screen *display*))
(defvar *root* (xlib:screen-root *screen*))

(defun show-window ()
  (let* ((black (xlib:screen-black-pixel *screen*))
         (white (xlib:screen-white-pixel *screen*))
         (window
           (xlib:create-window :parent *root* :x 0 :y 0 :width 640 :height 480
                               :class :input-output
                               :background white
                               :event-mask '(:key-press :key-release :exposure :button-
press
                                             :structure-notify)))
         (grackon (xlib:create-gcontext
                   :drawable window
    :foreground black
    :background white))
         (font (make-instance 'font :family "Times New Roman" :subfamily "Bold Italic"
                                    :size 12 :antialiased t)))
     (unwind-protect
         (progn
           (xlib:map-window window)
           (setf (xlib:gcontext-foreground grackon) black)
```

```
          (xlib:event-case (*display* :force-output-p t
                                      :discard-p t)
            (:exposure ()
                       (draw-text window grackon font "The quick brown fox jumps over
                       (when (= 0 (random 2))
                         (rotatef (xlib:gcontext-foreground grackon) (xlib:gcontext-
background grackon)))
                       (draw-text window grackon font "      ,   ." 100 (+ 100 (basel
to-baseline window font)))
                       (setf (font-antialiased font) (= 0 (random 2)))
                       (if (= 0 (random 2))
                           (setf (font-subfamily font) "Regular")
                           (setf (font-subfamily font) "Italic"))
                       (draw-text window grackon font "      ,     ' ." 100 (+ 100 (*
to-baseline window font))))
                       (draw-text window grackon font "Press space to exit.    ." 100
to-baseline window font)))))
             (:button-press () t)
             (:key-press (code state) (char= #\Space (xlib:keycode->character *dis-
play* code state)))))
      (progn
        (xlib:free-gcontext grackon)
        (xlib:destroy-window window)
        (xlib:display-force-output *display*)))))
```

Result is


[emacs] 0**slime-repl sbcl* 1-lisp@conference.jabber.ru 2+mc
The quick brown fox ju
Съешь же ещё этих м
Жебракують філософ
Press space to exit. Ha

3 Dictionary

`xft` [Package]

Package contains `api` for TrueType text rendering using `clx`, XRender. Glyphs information is obtained by `zpb-ttf`. Font rasterization is made by `cl-vectors`.

`*font-dirs*` [Variable]

List of directories, which contain TrueType fonts.

`cache-fonts` [Function]

Caches fonts from [Variable *font-dirs*], page 6 directories.

`cache-font-file` *pathname* [Function]

Caches font file.

`get-font-families` [Function]

Returns cached font families.

`get-font-subfamilies` *font-family* [Function]

Returns font subfamilies for current *font-family*. For e.g. regular, italic, bold, etc.

`font` [Class]

Class precedence list: `font, standard-object, t`

Slots:

- `family` — initarg: `:family`; reader: `clx-truetype:font-family`; writer: `(setf clx-truetype:font-family)`

  Font family.

- `subfamily` — initarg: `:subfamily`; reader: `clx-truetype:font-subfamily`; writer: `(setf clx-truetype:font-subfamily)`

  Font subfamily. For e.g. regular, italic, bold, bold italib.

- `size` — initarg: `:size`; reader: `clx-truetype:font-size`; writer: `(setf clx-truetype:font-size)`

  Font size in points.

- `underline` — initarg: `:underline`; reader: `clx-truetype:font-underline`; writer: `(setf clx-truetype:font-underline)`

  Draw line under text string.

- `strikethrough` — initarg: `:strikethrough`; reader: `clx-truetype:font-strikethrough`; writer: `(setf clx-truetype:font-strikethrough)`

  Draw strike through text string.

- `overline` — initarg: `:overline`; reader: `clx-truetype:font-overline`; writer: `(setf clx-truetype:font-overline)`

  Draw line over text string.

- `background` — initarg: `:background`; reader: `clx-truetype:font-background`; writer: `(setf clx-truetype:font-background)`

  Background color.

- `foreground` — initarg: `:foreground`; reader: `clx-truetype::font-foreground`; writer: `(setf clx-truetype::font-foreground)`

  Foreground color.
- `overwrite-gcontext` — initarg: `clx-truetype::overwrite-gcontext`; reader: `clx-truetype:font-overwrite-gcontext`; writer: `(setf clx-truetype:font-overwrite-gconte`

  Use font values for background and foreground colors.
- `antialias` — initarg: `clx-truetype::antialias`; reader: `clx-truetype:font-antialias`; writer: `(setf clx-truetype:font-antialias)`

  Antialias text string.

Class for representing font information.

`font-equal` *font1 font2* [Generic Function]
Returns t if two font objects are equal, else returns nil.

`screen-default-dpi` *screen* [Function]
Returns default dpi for *screen*. pixel width * 25.4/millimeters width

`screen-dpi` *screen* [Function]
Returns current dpi for *screen*.

`(setf screen-dpi)` *value screen* [Function]
Sets current dpi for *screen*.

`font-ascent` *drawable font* [Function]
Returns ascent of *font*. *drawable* must be window, pixmap or screen.

`font-descent` *drawable font* [Function]
Returns descent of *font*. *drawable* must be window, pixmap or screen.

`font-line-gap` *drawable font* [Function]
Returns line gap of *font*. *drawable* must be window, pixmap or screen.

`baseline-to-baseline` *drawable font* [Function]
Returns distance between baselines of *font*. *drawable* must be window, pixmap or screen. ascent - descent + line gap

`text-bounding-box` *drawable font string* [Function]
Returns text bounding box. *drawable* must be window, pixmap or screen. Text bounding box is only for contours. Bounding box for space (#x20) is zero.

`text-width` *drawable font string* [Function]
Returns width of text bounding box. *drawable* must be window, pixmap or screen.

`text-height` *drawable font string* [Function]
Returns height of text bounding box. *drawable* must be window, pixmap or screen.

`text-line-bounding-box` *drawable font string* [Function]
Returns text line bounding box. *drawable* must be window, pixmap or screen. Text line bounding box is bigger than text bounding box. It's height is ascent + descent, width is sum of advance widths minus sum of kernings.

`text-line-width` *drawable font string* [Function]
Returns width of text line bounding box. *drawable* must be window, pixmap or screen. It is sum of advance widths minus sum of kernings.

`text-line-height` *drawable font string* [Function]
Returns height of text line bounding box. *drawable* must be window, pixmap or screen.

`xmin` *bounding-box* [Function]
Returns left side x of *bounding-box*

`ymin` *bounding-box* [Function]
Returns bottom side y of *bounding-box*

`xmax` *bounding-box* [Function]
Returns right side x of *bounding-box*

`ymax` *bounding-box* [Function]
Returns top side y of *bounding-box*

`draw-text` *drawable gcontext font string x y &key start end* [Function]
Draws text string using *font* on *drawable* with graphic context *gcontext*. *x*, *y* are the left point of base line. *start* and *end* are used for substring rendering. If *gcontext* has background color, text bounding box will be filled with it. Text line bounding box is bigger than text bounding box. *drawable* must be window or pixmap.

`draw-text-line` *drawable gcontext font string x y &key start end* [Function]
Draws text string using *font* on *drawable* with graphic context *gcontext*. *x*, *y* are the left point of base line. *start* and *end* are used for substring rendering. If *gcontext* has background color, text line bounding box will be filled with it. Text line bounding box is bigger than text bounding box. *drawable* must be window or pixmap.

`font-lines-height` *drawable font lines-count* [Function]
Returns text lines height in pixels. For one line height is ascender+descender. For more than one line height is ascender+descender+linegap.

Appendix A Concept Index

(Index is nonexistent)

Appendix B Function Index

Appendix C Variable Index

F

X

Appendix D Type Index

Colophon

This manual is maintained in Texinfo, and automatically translated into other forms (e.g. HTML or pdf). If you're *reading* this manual in one of these non-Texinfo translated forms, that's fine, but if you want to *modify* this manual, you are strongly advised to seek out a Texinfo version and modify that instead of modifying a translated version.