

CLX-TRUETYPE

CLX TrueType Font Renderer

## Table of Contents

Introduction .....	1
1 Overview .....	2
2 Examples .....	3
3 Dictionary .....	6
Appendix A Concept Index .....	9
Appendix B Function Index.....	10
Appendix C Variable Index .....	11
Appendix D Type Index.....	12
Colophon.....	13

## 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
```

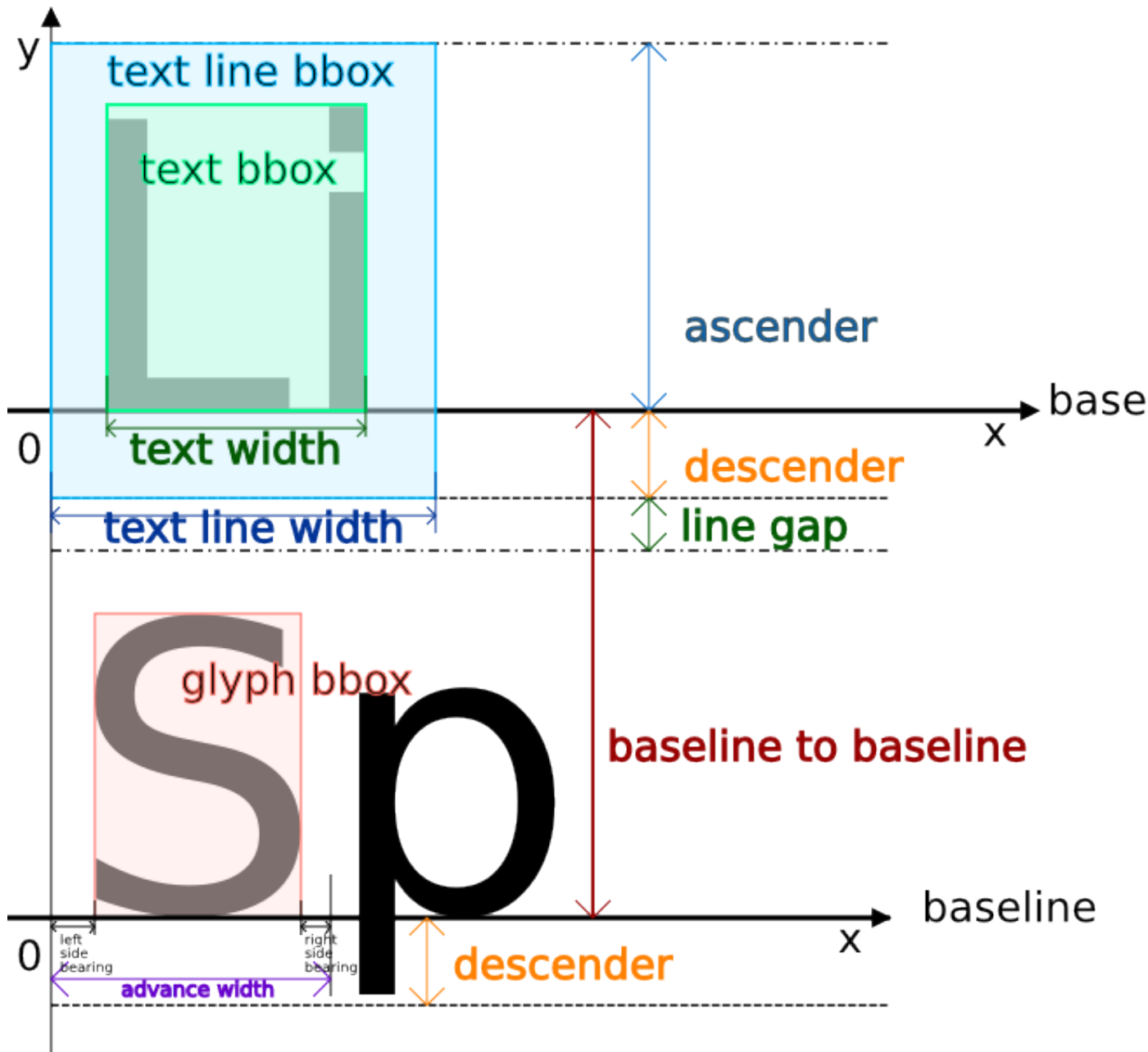
will get you a local copy.

<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:destroy-window window)
    (xlib:close-display *display*)))
```

```

(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 jumps over the lazy dog.  
Съешь же ещё этих мягких французских  
Жебракуютъ філософъ.  
Press space to exit. На

### 3 Dictionary

<b>xft</b>	[Package]
Package contains <b>api</b> for TrueType text rendering using <b>clx</b> , <b>XRender</b> . Glyphs information is obtained by <b>zpb-ttf</b> . Font rasterization is made by <b>cl-vectors</b> .	
<b>*font-dirs*</b>	[Variable]
List of directories, which contain TrueType fonts.	
<b>cache-fonts</b>	[Function]
Caches fonts from <b>*font-dirs*</b> directories.	
<b>cache-font-file</b> <i>pathname</i>	[Function]
Caches font file into hashmap.	
<b>get-font-families</b>	[Function]
Returns cached font families.	
<b>get-font-subfamilies</b> <i>font-family</i>	[Function]
Returns font subfamilies for current. For e.g. regular, italic, bold, etc.	
<b>font</b>	[Class]
Class precedence list: <b>font</b> , <b>standard-object</b> , <b>t</b>	
Slots:	
<ul style="list-style-type: none"> <li>• <b>family</b> — initarg: <b>:family</b>; reader: <b>clx-truetype:font-family</b>; writer: <b>(setf clx-truetype:font-family)</b> Font family.</li> <li>• <b>subfamily</b> — initarg: <b>:subfamily</b>; reader: <b>clx-truetype:font-subfamily</b>; writer: <b>(setf clx-truetype:font-subfamily)</b> Font subfamily. For e.g. regular, italic, bold, bold italic.</li> <li>• <b>size</b> — initarg: <b>:size</b>; reader: <b>clx-truetype:font-size</b>; writer: <b>(setf clx-truetype:font-size)</b> Font size in points.</li> <li>• <b>underline</b> — initarg: <b>:underline</b>; reader: <b>clx-truetype:font-underline</b>; writer: <b>(setf clx-truetype:font-underline)</b> Draw line under text string.</li> <li>• <b>strikethrough</b> — initarg: <b>:strikethrough</b>; reader: <b>clx-truetype:font-strikethrough</b>; writer: <b>(setf clx-truetype:font-strikethrough)</b> Draw strike through text string.</li> <li>• <b>overline</b> — initarg: <b>:overline</b>; reader: <b>clx-truetype:font-overline</b>; writer: <b>(setf clx-truetype:font-overline)</b> Draw line over text string.</li> <li>• <b>background</b> — initarg: <b>:background</b>; reader: <b>clx-truetype:font-background</b>; writer: <b>(setf clx-truetype:font-background)</b> Background color.</li> </ul>	



- **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-gcontext)`  
Use font values for background and foreground colors.
- **antialiased** — initarg: `clx-truetype::antialiased`; reader: `clx-truetype:font-antialiased`; writer: `(setf clx-truetype:font-antialiased)`  
Antialias text string.

Class for representing font information.

**screen-default-dpi** *screen* [Function]  
Returns default dpi for @var{screen}. pixel width \* 25.4/millimeters width

**screen-dpi** *screen* [Function]  
Returns current dpi for @var{screen}.

**(setf screen-dpi)** *value screen* [Function]  
Sets current dpi for @var{screen}.

**font-ascent** *drawable font* [Function]  
Returns ascent of @var{font}. @{drawable} must be window, pixmap or screen.

**font-descent** *drawable font* [Function]  
Returns descent of @var{font}. @{drawable} must be window, pixmap or screen.

**font-line-gap** *drawable font* [Function]  
Returns line gap of @var{font}. @{drawable} must be window, pixmap or screen.

**baseline-to-baseline** *drawable font* [Function]  
Returns distance between baselines of @var{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. @var{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. @var{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. @var{drawable} must be window, pixmap or screen.
- xmin** *bounding-box* [Function]  
Returns left side x of @var{bounding-box}
- ymin** *bounding-box* [Function]  
Returns bottom side y of @var{bounding-box}
- xmax** *bounding-box* [Function]  
Returns right side x of @var{bounding-box}
- ymax** *bounding-box* [Function]  
Returns top side y of @var{bounding-box}
- draw-text** *drawable gcontext font string x y &key start end* [Function]  
Draws text string using @var{font} on @var{drawable} with graphic context @var{gcontext}. @var{x}, @var{y} are the left point of base line. @var{start} and @var{end} are used for substring rendering. If @var{gcontext} has background color, text bounding box will be filled with it. Text line bounding box is bigger than text bounding box. @var{drawable} must be window or pixmap.
- draw-text-line** *drawable gcontext font string x y &key start end* [Function]  
Draws text string using @var{font} on @var{drawable} with graphic context @var{gcontext}. @var{x}, @var{y} are the left point of base line. @var{start} and @var{end} are used for substring rendering. If @var{gcontext} has background color, text line bounding box will be filled with it. Text line bounding box is bigger than text bounding box. @var{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

### B

baseline-to-baseline ..... 7

### C

cache-font-file ..... 6

cache-fonts ..... 6

### D

draw-text ..... 8

draw-text-line ..... 8

### F

font-ascent ..... 7

font-descent ..... 7

font-line-gap ..... 7

font-lines-height ..... 8

### G

get-font-families ..... 6

get-font-subfamilies ..... 6

### S

screen-default-dpi ..... 7

(setf screen-dpi) ..... 7

### T

text-bounding-box ..... 7

text-height ..... 7

text-line-bounding-box ..... 7

text-line-height ..... 8

text-line-width ..... 8

text-width ..... 7

### X

xmax ..... 8

xmin ..... 8

### Y

ymax ..... 8

ymin ..... 8

Appendix C Variable Index

<b>F</b>	<b>X</b>
*font-dirs*..... 6	xft..... 6

Appendix D Type Index

font..... 6

## 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.