Fonts Layout Rendering

Contents

FreeType Font Rendering

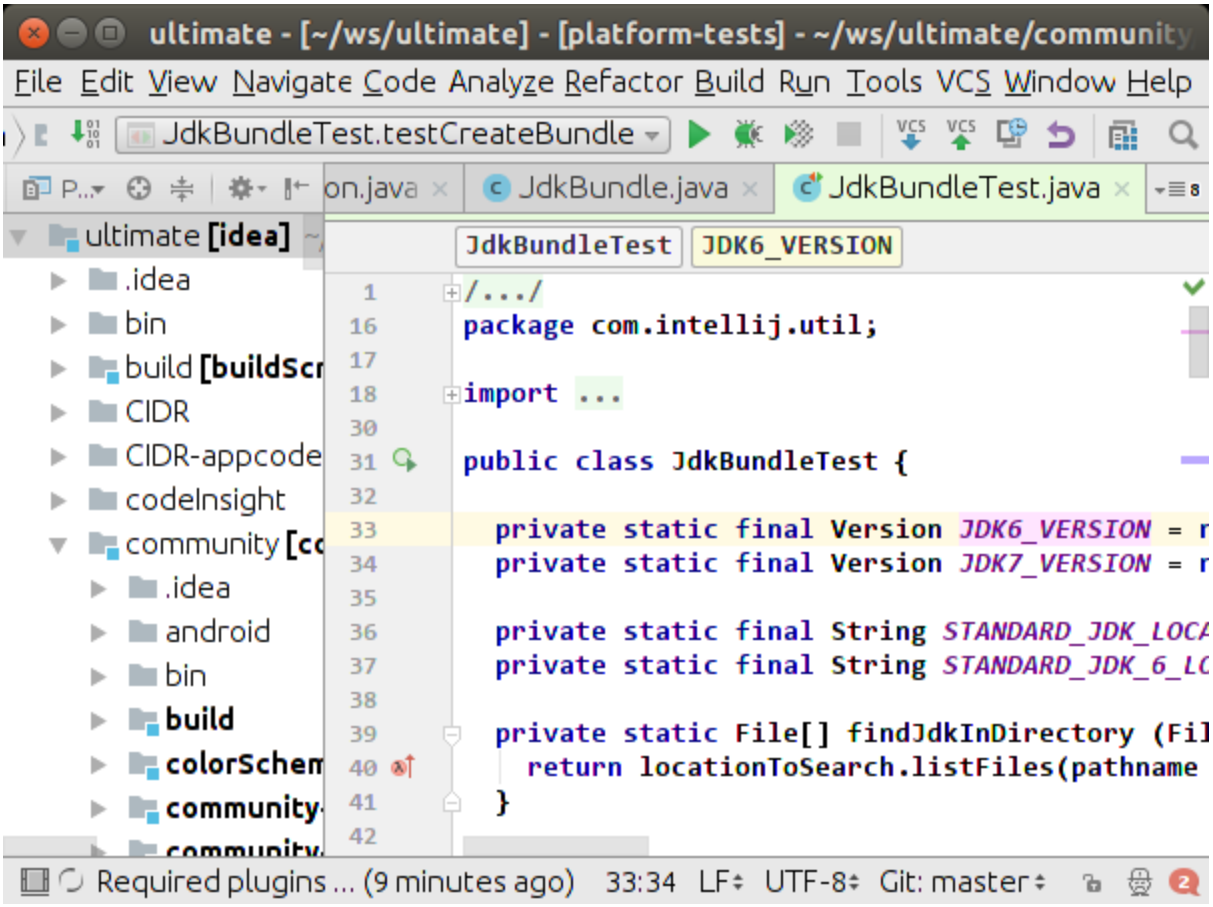
OpenJDK uses the FreeType library for font rendering on Linux (and even on Windows and macOS for some special rendering modes). Freetype is used for generating glyph images and provide metrics from font files.

Originally OpenJDK uses the Fontconfig library only for retrieving font directories on particular Unix (Solaris, Linux) platforms. However, on modern Unix desktops, Fontconfig is also used as a source of rendering hints which are specific for the particular font. These hints can be directly mapped to the rendering modes of FreeType library.

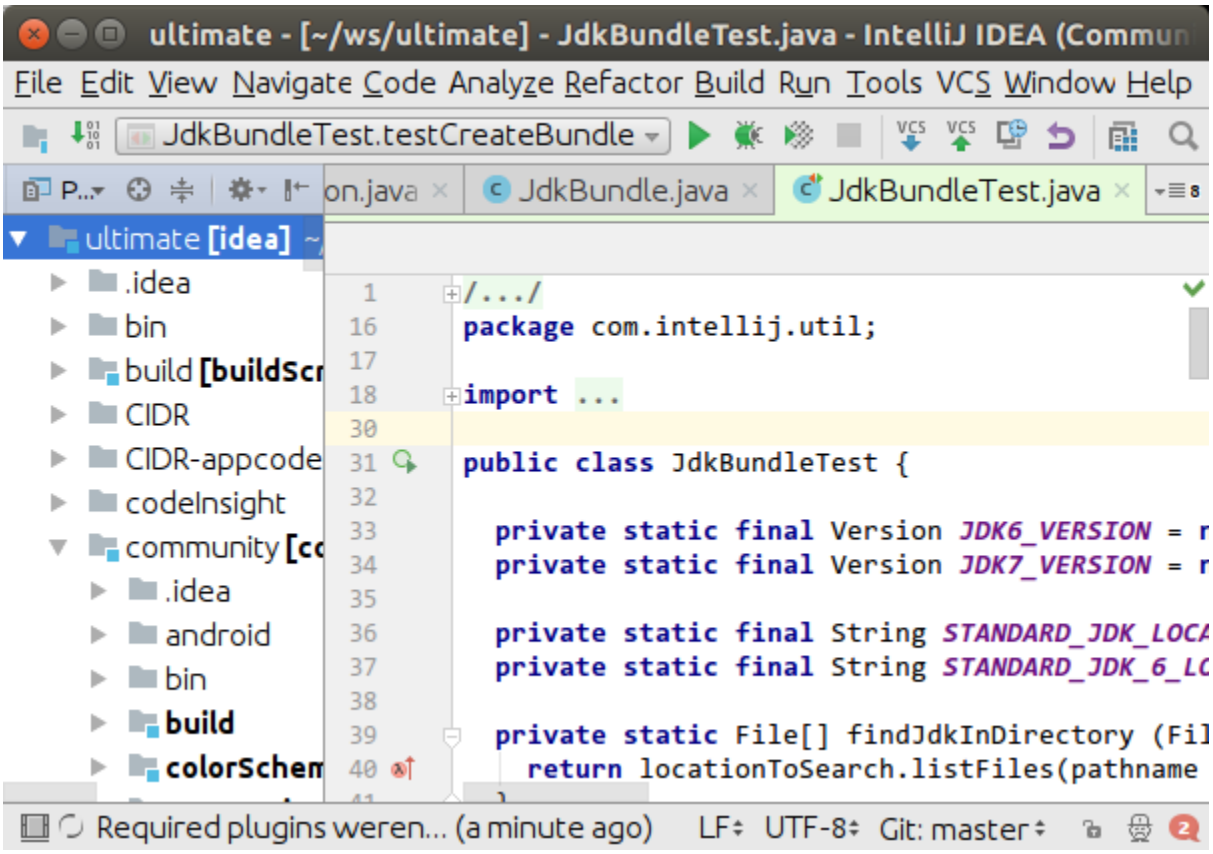
This approach was initially implemented by the open source community within openjdk-fontfix patch. Patched JDK demonstrated much better rendering quality. However, it was not quite complete and reliable solution for bundling with standalone JDK distribution, also the patch didn’t play well with Java2D rendering hints. So, we decided to implement our own solution based on Fontconfig in Jetbrains Runtime.

Also, for historical reasons OpenJDK uses 72dpi rendering which usually doesn't correspond with actual screen dpi. That leads to some visual artifacts in font rendering. In JetBrains Runtime we tried to resolve this problem by providing actual dpi for FreeType library. The only problem is that glyphs rendered with higher dpi have large sizes, so java desktop applications unexpectedly receive rendered strings with larger sizes. So, for backward compatibility, we adjust font sizes passed to FreeType library to compensate glyphs increase.

Initial implementation of our solution provided the natural way of rendering fonts. However, quite a few users were unhappy because of increased thickness in the new font rendering especially it was noticeable in low-resolution display modes which are quite common for Linux notebooks. So, we decided to tweak platform provided hints in order to make font rendering of small fonts more like original JDK font rendering. It was easily implemented by bundling our own font.conf file in Fontconfig library format.



JetBrains OpenJDK (build 1.8.0_112-release-752-b4)



Font layout

JetBrains Runtime uses HarfBuzz font layout engine, backported from OpenJDK 9, where it replaced legacy ICU engine. It provides a better support for fonts containing custom layout rules (such rules are used for complex writing systems and e.g. for implementing ligatures). Additionally, JetBrains Runtime optimizes usage of HarfBuzz API (creating hb_face_t object only once per font face, and not per each text layout request), which yields a significant performance improvement when working with fonts containing a very large number of layout rules (e.g. a popular Fira Code font).

Emoji support on macOS

JetBrains Runtime supports rendering of color emoji glyphs on macOS. Technically, this is achieved by using a newer native API to render such glyphs (CTFontDrawGlyphs call is used, while OpenJDK uses deprecated CGContextShowGlyphsAtPoint), while also making rendering infrastructure understand a new format of cached glyph image - colored image with transparency, which is blitted directly, not using set foreground color in any way.

Text rendering on 4k monitors (OSX)

The main reason for performance degradation in the editor on IDEA platform based products on OSX with 4k monitors is the huge amount of glyphs that need to be rendered. Also, because of details of implementation of rendering pipeline on OSX subpixel AA rendering is several times slower than grayscale one. That was the reason of disabling subpixel rendering in Oracle JDK. We've enabled this feature in the JetBrains Runtime based on OpenJDK8 code to provide the same visual experience that users have in native applications. It was acceptable for handling text content at that time. However, rendering huge amount of glyph on 4k monitors is quite slow in subpixel mode. We've performed some investigation and speeded things up but still have about 2X performance drop in subpixel rendering comparing with grayscale AA.

So if you have performance problems with scrolling on IDEA, you can switch to grayscale AA (Preferences->Appearance->Antialiasing)