Pregenerating Text Data Files

PAGE CONTENTS

By default, app text is rendered using the source fonts. To improve performance, you can pregenerate text data files for the fonts used in your app. Pregenerating text data files has the greatest impact on the performance cost of first drawing text, most often incurred when first displaying a page or scene. On lower-end platforms, text data pregeneration can make first-render of text up to 2.5 times faster.

The Text Pregenerator tool is supported on macOS, Windows, and Linux.

See also Performance Best Practices for You.i Engine One Apps, including the section on Optimizing Text Performance.

Running the Text Pregenerator

From the `/tools/TextPregenerator` folder of your Engine installation, run:

```
./TextPregenerator --font path/to/my/font.ttf
```

For example:

```
./TextPregenerator --font ../../RNSampleApp/youi/AE/assets/yi_Gotham-Bold.ttf
```

You'll see output that looks something like this:

```
2020-10-20 12:10:40.388 TextPregenerator[14278:337106] C/:   --- LOG START ---
119 codepoints selected.
Processing font file ../../RNSampleApp/youi/AE/assets/yi_Gotham-Bold.ttf...120.9 KiB bytes written
to '../../RNSampleApp/youi/AE/assets/yi_Gotham-Bold.ttf.pregen'.
Done after 0.1 seconds. 119 bitmaps generated for 119 unique glyphs.
```

By default, the file `<font-file-name>.pregen` is output to the same location as the font file. Files with the `.pregen` extension are automatically loaded by the Engine. You can change the file name and location with the `--data <file>` option.

You can use `--help` to learn about all of the available options, but we call out a few of the more commonly used options below.

Loading .pregen Files at Build Time

Pregenerated text data files are loaded automatically by the Engine if they're found in the same folder as the associated font files. In mos cases, pregenerated text data files placed in your app's `AE/assets` folder are also loaded automatically.

In cases where fonts are being loaded from memory (for example, when fonts are downloaded from the internet), you can load a pregenerated text data file manually using the CYITextEngine::AddPregeneratedTextData function.

If loading of the `.pregen` file during app build fails for any reason, a warning appears in the build log.

Since `.pregen` files act as a sort of cache for the actual font files, the font files still need to be included in the app's `AE/assets/` folder.

Including a Custom Set of Glyphs

By default, a common subset of glyphs present within the file are pregenerated. That subset contains:

- all uppercase and lowercase letters of the Latin alphabet
- numbers
- some punctuation and symbol glyphs

To include a different set of glyphs in the pregenerated text data file, use the `--codepoints` and `--characters` options.

You can also pass a "training" document as input to the tool using the `--training-document` option. With this option, only characters found in that document are pregenerated. Each character in the document is added to the characters list. The document must be UTF-8 encoded.

Use the Dev Panel's Device Information widget to see how many glyph requests have used the `.pregen` file and how many requests have fallen back to the original font. Use this information to determine whether to expand the set of glyphs included in the `.pregen` file.

NOTE

Including all glyphs from your fonts (with the `--codepoints everything` option) can result in large `.pregen` files and an increased final application package size.

Pregenerate Bitmap Data Instead of SDF

By default, the tool pregenerates glyphs for use with SDF (signed distance field) text scene nodes and pregenerates all styles found in the font. You can also pregenerate plain bitmap data with `--type`, and you can specify which styles to pregenerate with `--style`.

Android Apps with Custom build.gradle Files

In order to be performant on Android, `.pregen` files must be stored in the application's `.apk` in an uncompressed form. The Engine handles this by default, but applications that use a custom `build.gradle` file must add the `.pregen` extension to the `noCompress` list. Note that because `.pregen` files internally compress their data, disabling compression on `.pregen` files does not significantly impact the final size of the `.apk`.