Font Asset and Editor

Overview of the Font Asset and Editor

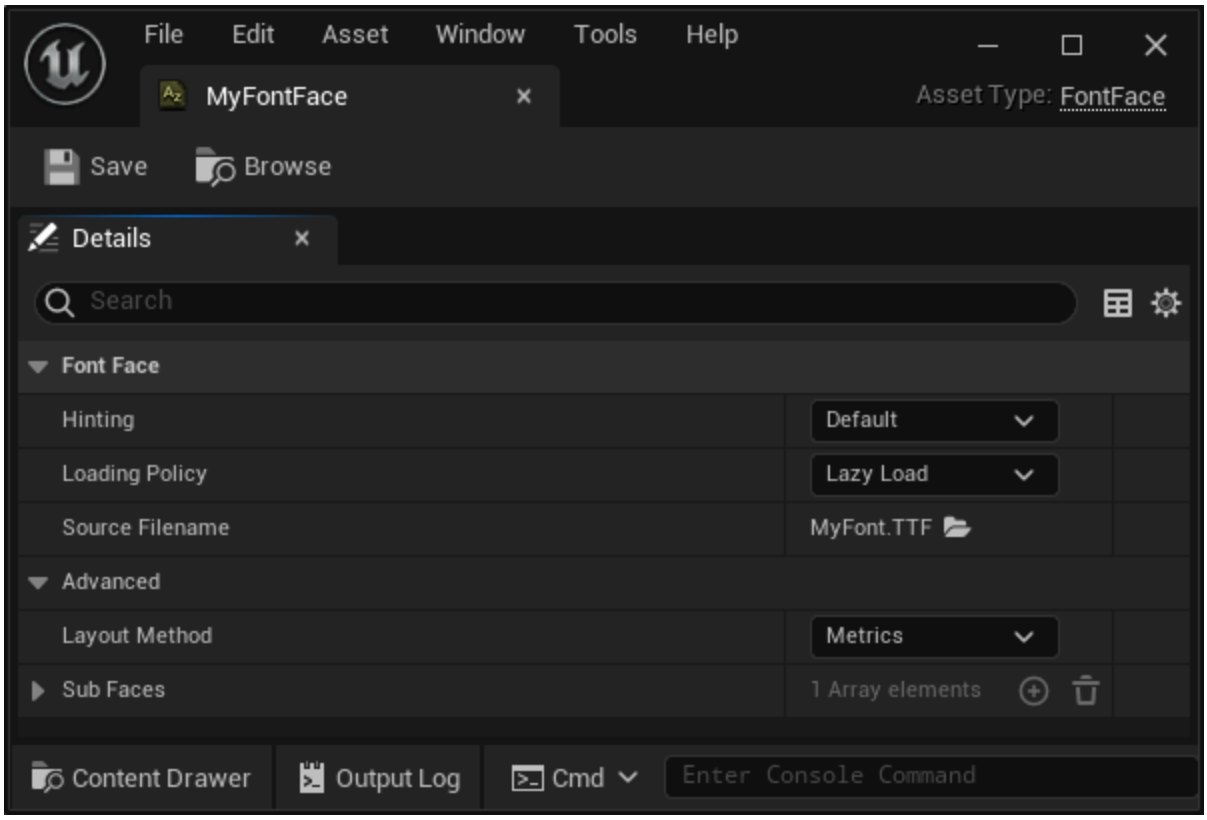
This page covers the **Font** and **Font Face** asset types that can be used with the **Font Editor**.

Font Assets

Fonts in Unreal Engine are categorized as a **Font** asset and use two caching methods, **Runtime** which is in the form of a Composite Font or **Offline** which is the older pre-computed Font Atlas method. You can switch between the two methods by opening up a Font asset in the Font Editor (this provides a simple way to convert existing Font assets from Offline to the new composite method without having to replace them).

Font Face Assets

The **Font Face** asset is created when you import a font and it stores the font data that can be referenced by the Font asset. This means that the same font data can be reused across multiple Font assets or even with multiple typefaces within the asset, and ultimately reduces memory consumption.



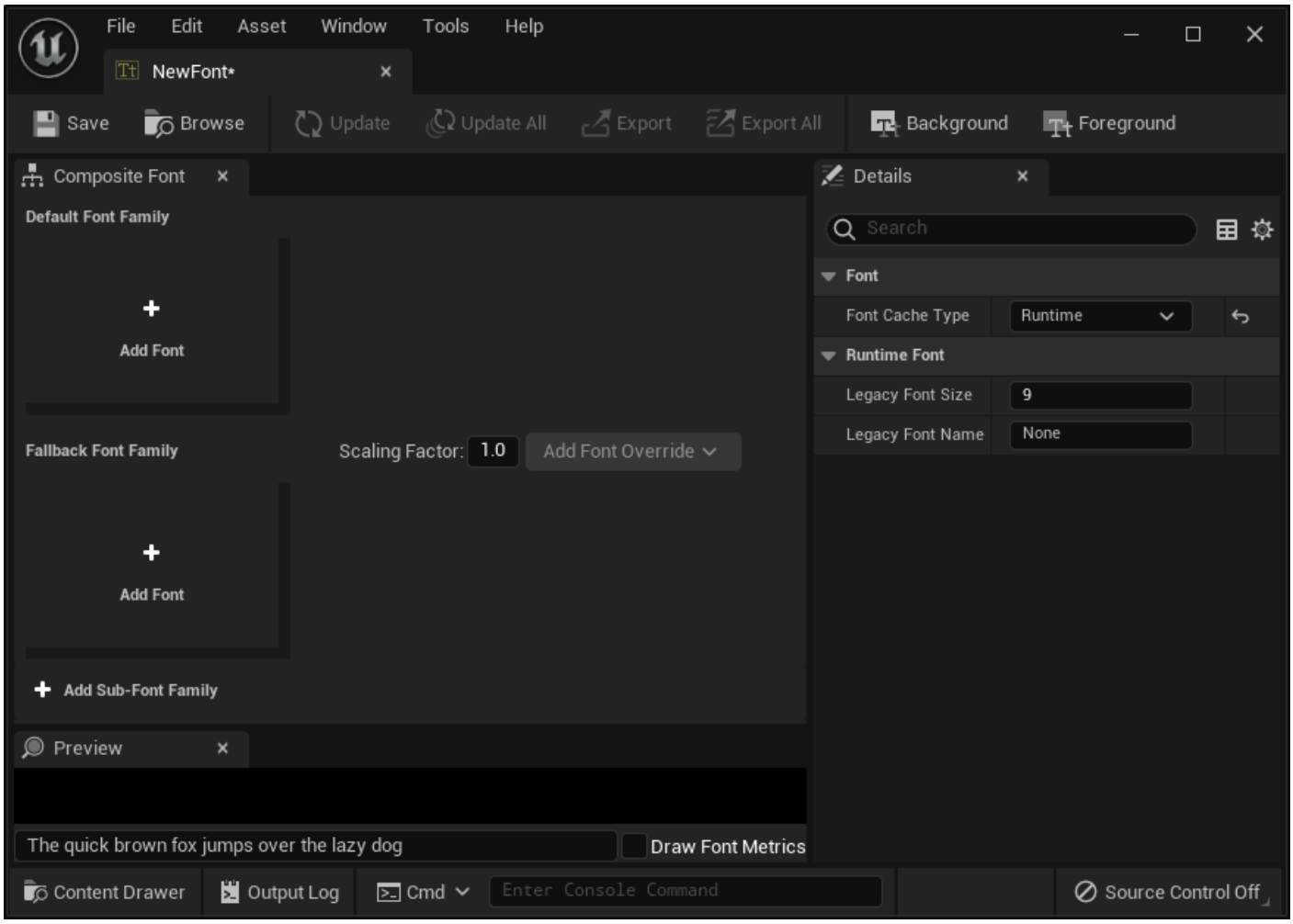
When you open your Font Face Asset, you have access to the settings for **Hinting** and **Loading Policy**.

Property	Description
Hinting	The hinting algorithm to use with the font face: • **Default**: Use the default hinting specified in the font. • **Auto**: Force the use of an automatic hinting algorithm. • **Auto Light**: Force the use of an automatic hinting algorithm, optimized for non-monochrome displays. • **Monochrome**: Force the use of an automatic hinting algorithm optimized for monochrome displays. • **None**: Do not use hinting.
Loading Policy	Enum controlling how this font face should be loaded at runtime. See the enum for more explanations of the options: • **Lazy Load**: Lazy load the entire font into memory. This will consume more memory than Streaming, however, there will be zero file-IO when rendering glyphs within the font, although the initial load may cause a hitch. • **Stream**: Stream the font from disk. This will consume less memory than Lazy Load or Inline, however, there will be file-IO when

Property	Description
	rendering glyphs, which may cause hitches under certain circumstances or on certain platforms. • **Inline**: Embed the font data within the asset. This will consume more memory than Streaming, however it is guaranteed to be hitch free (only valid for font data within a Font Face asset).
Source File Name	The filename of the font face we were created from. This may not always exist on disk, as we may have previously loaded and cached the font data inside this asset.
Layout Method	This selects the method to use when laying out the font. Try changing this if you notice clipping or height issues with your font: • **Metrics**: Layout the font using metrics data available in the font. This is typically the desired option, however some fonts have broken or incorrect metrics and may yield better results by using the bounding box values to layout the font. • **Bounding Box**: Layout the font using the values from its bounding box. This typically yields a larger line height for fonts that have valid metrics, however it can also produce much better results for fonts that have broken or incorrect metrics.
Sub Faces	Transient cache of the sub-faces available within this face.

Font Editor

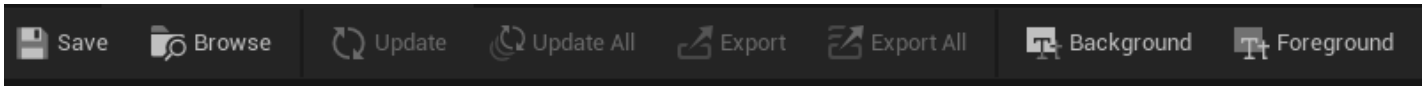
When you double-click on a Font asset in the **Content Browser**, it will open up inside of the **Font Editor** window.



Click image for a full view.

A breakdown of the Font Editor Window is presented below:

Toolbar Menu



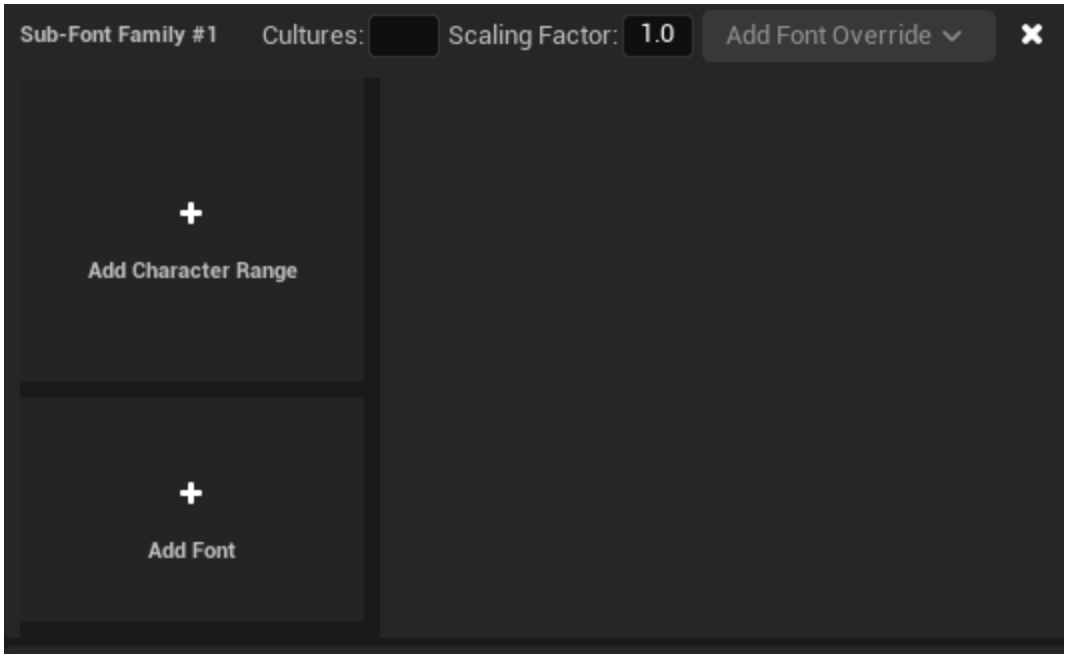
From this menu, you can save any changes you make, find the asset in the **Content Browser**, change the Background Color of the preview window or the Foreground Color (text color) in the preview window. There are options for Updating or Exporting changes being made, however, these options are only available within the **Offline** cache mode.

Default Font Family

In this window, you can assign the Default Font Family for use with this Font asset. You can add versions of a particular Font style (for example Normal, Bold, Italics, Underline, etc.) or have a collection of different Font styles as one Composite Font. If you have created a blank Font asset, you can assign a font from inside this window as well.

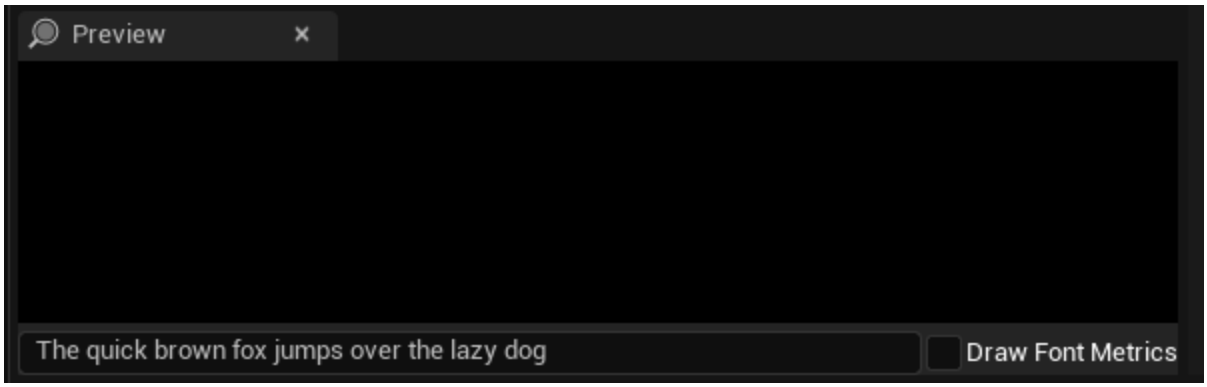
Sub-Font Family

In this window, when you click the **Add Sub-Font Family** button, you can assign the Sub-Font Family for this Font asset to use.



Here, you can specify a Character Range, and if a character entered falls within the range, you can specify a different Font style to use instead of the Default. This is useful for when you want to use different Font types for different languages.

Preview



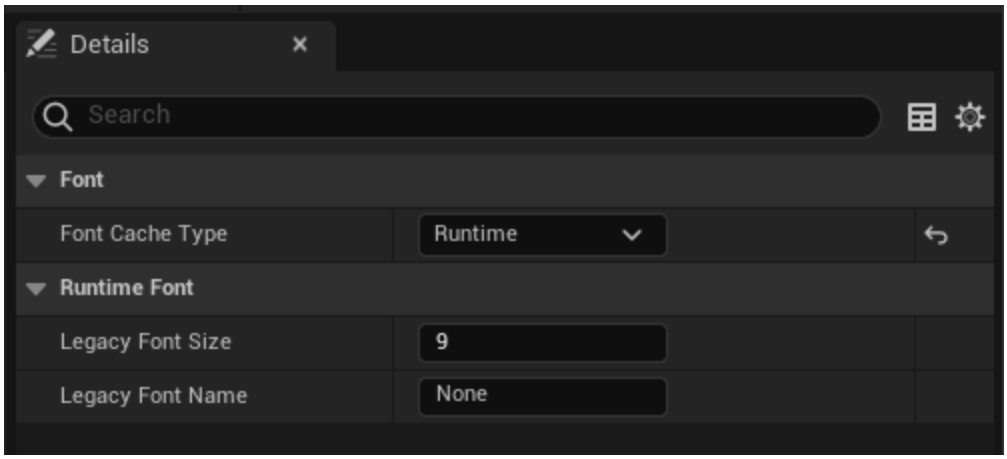
This window allows you to preview your fonts and provides a text entry box for entering sample texts.

Draw Font Metrics

The **Draw Font Metrics** toggle will overlay the line height, glyph bounding boxes, and base-line as part of the preview.

- **Base Line** - This is the line in which the text sits.
- **Line Bounds** - This is the bounding box created for the length of the given text string.
- **Grapheme Cluster Bounds** - This is the bounding box drawn around what is considered a logical character in a given language, and may be comprised of several glyphs (for example, a base character and accent glyph).
- **Glyph Bounds** - This is the bounding box drawn around the given glyph.

Details

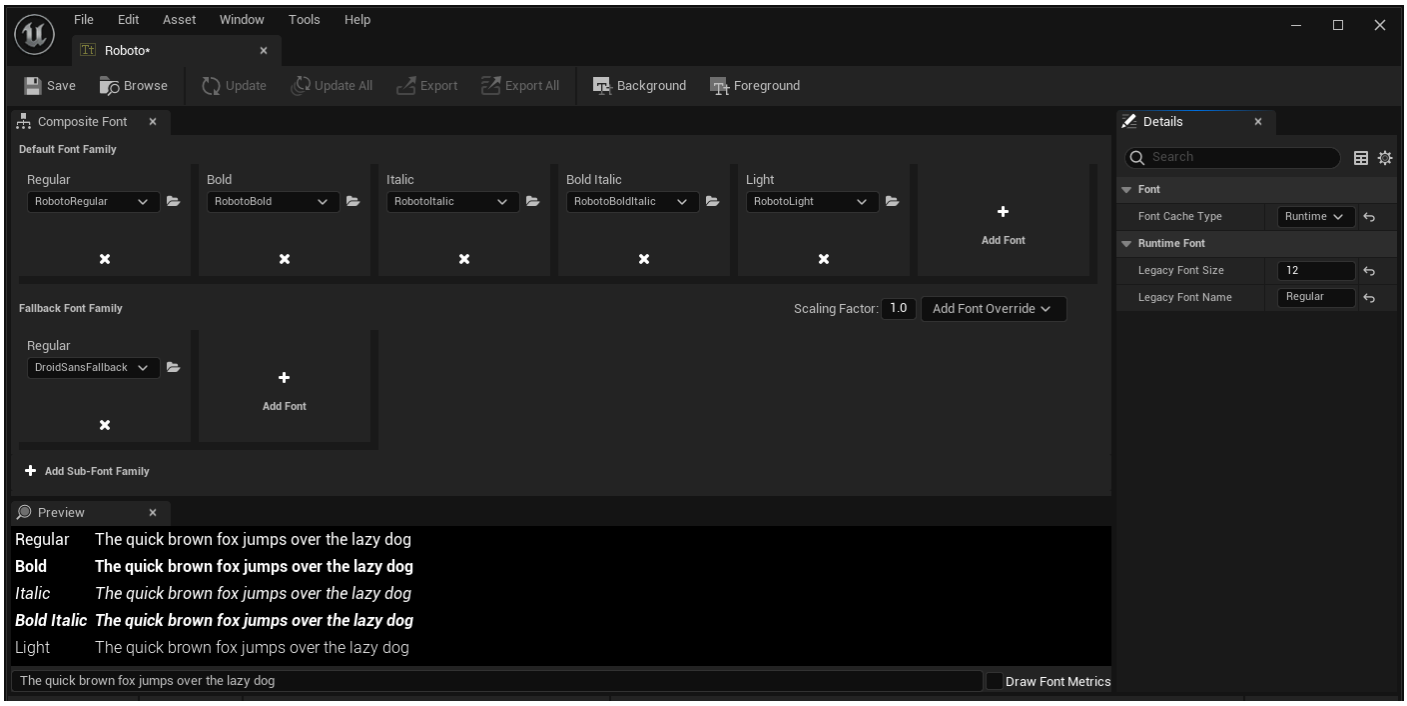


In this window, you can change the Font Cache Type as well as change the Font Size and Font Name (for Runtime).

- If you are using the older method, you can still change the parameters for your Font while in Offline cache mode.
- You can also convert any existing Font assets from **Offline** to **Runtime** without having to replace them.

Example Font Asset

An example Font asset is shown below.



Click image for a full view.

A Composite Font will always contain a Default Font Family, and may also contain any number of Sub-Font Families that should be used for a given range of characters. Each Font Family is itself made up of any number of Font Faces that can be named based on their style. At runtime, the most suitable Font to use for each character (based on the Fonts available) in the Font Family for that character range is used.

As seen in the example image above, the Japanese text falls within the character ranges of the Japanese font family, and therefore, is drawn using Source Han Sans rather than the Default Font Family (Roboto). Fonts in a Sub-Font Family are preferably chosen by name match, as in the case of Regular, Bold, and Light, however they can also fallback to matching based on the attributes of the Default Font, as is the case with Bold Italic (it automatically chose the Bold Japanese font because the font contained the Bold attribute).