Where does the resolution, aka units per em, of the em squares of fonts come from? Type designers may design on paper, film, Illustrator or in the sand, but eventually, if a font's going to be widely used today, it must be digital. All digital fonts are defined on a grid described by the number of vertical units I. The en. Most fonts are defined on 1,000 such units per em, which for many or most fonts is fine.

So, starting with a brief description of what happens to the units per em value of a font in OS font software, like that in most an OSs, then describe what kinds of fonts need more than 1,000 units, and finally some history of units per em and links to current specs.

What are upm used for?
A font with a 1,000 unit em, set at 12 point and rendered on a 300 dpi device, goes through at least this transformation; a 12 point 300 dpi em is 50 pixels, so a 1,000 unit per em font is scaled to 1/20th of its original size and its glyphs can then be rendered on an em that's 50 units tall. A font with units per on of 2048, would be scaled to 1/40.96ths and so on.

Though this straight-forward translation from upm to pixels/inches is current in most print environments, modern OSs do play all kinds of tricks with this simple scaling, as does CSS. The former has examples of scaling the units per em to 3 times the final pixels per em, and then rendering in 1/3rds so rgb pixels can be calculated for anti-aliased font display. The latter as never accurately defined typographic points on the screen, preferring a complex of possible solutions relying on an assumed user distance, (from all devices), and an approximation of actual size on screen based on quasi-accurate dpi values.

Despite what happens to the mapping of upm to pixels and size, the units per em relative to the detail in the font is still entirely in the type developer's hands. I wish the units per em decision was as simple as this; if your design does not have any details of shapes and spaces that are smaller the 1/1000ths, that's the best upm value to use. That's what I thought until I ran into an example in my own work, (vimeo.com/420680703), where the counter space is 2/1000, but I was unable to locate a point in non-orthogonal space that properly captured that detail among the stem weights. That discovery stopped work on this design until I could scale the em to 2000 and evaluate the work required to fix the artifacts sure to appear from scaling the upm.

Background
My experience with upm started designing fonts in 1978 on a 10", 100% cotton rag paper em, with a thin pencil line around 100th of an inch. This lent an effective drawing resolution of 1000 pencil lines per em which was then converted to a 10" film copy of the drawing, scanned at 8,000 dpi, (80,000 units per em), to form a very high resolution digital bitmap image of the glyphs, as outline fonts were to hard to scale at the time.

These scans were then scaled to master bitmaps for various type size, which after manual editing on alpha-numeric computer terminals cleaned them up so the hi resolution bitmap fonts could be made into products for early digital typesetters. By 1980 the paper and film had been augmented by a computer-aided design system, Ikarus, that used an 8,640 unit em to define digital outlines that could replace the scanning and scaling of bitmaps, and present the final bitmap masters for the typesetter from scaled outlines.

By 1982, typesetters were beginning to use digital outlines directly, doing the scaling of glyphs and rendering bitmaps on-the-fly in "imagesetters" used in print preproduction. Working for a second company, I was part of the development of an interactive type design system based on a 4,320 unit per em mastering process.

Understanding that a lower units per em of a font takes up less file space than a higher unit per em font, and that the em did not have to scale anymore between the type designer's data and the final scaleable font products, in 1986 Adobe systems released the postscript type 3 font spec which defined the em at 1,000 units. And with the democratization of type design tools defaulting to 1,000 units per em, and many products unable to deal with units per em other than 1,000 for any postscript fonts, by 1990 many type designers thought 1,000 unit ems were all there ever was.

Someone was interested in this issue when Truetype was being defined, because the specification for that format allows a developer-defined unit per em from 64 through 16384. Microsoft defines the TrueType units per em as having a maximum positive value of 16383, and Freetype defines a unit per em range with a maximum positive value of 32767. The latter is not clear about a minimum, but gives the example of a 4 unit em with 25 possible point locations, (something I can't vouch for).

In all cases a power of two is recommended as performing better, though there is no proof that such performance gain actually exists today.

developer.apple.com/fonts/TrueType-Reference-Manual/RM06/Chap6head.html

docs.microsoft.com/en-us/typography/opentype/spec/ttch01

www.freetype.org/freetype2/docs/glyphs/glyphs-2.html.

So, based on that understanding, the designer of a single style, is advised to select a value that adequately represents the smallest detail of that style without compromise. The designer of a typeface family, including fonts with a wide variety of detail, may wish to choose different unit per em values based on the smallest detail requirements of each style. But the designer of a variable font family, only has one choice of units per em for all the design space of variable font.*

*And while a variable font developer may make an unregistered axis where the only change is that the contours round to a range of upm, the upm of the variable font that contains such an axis, must be the highest value of that upm range or higher. This has essentially been proposed as a pixel per em axis, but has so far been rejected by the owners of the spec.