The TrueType font engine is the software that convert the information in a TrueType font into a raster image suitable for display on screen or printer.

Step to rasterize a glyph into screen:

+ Scale the master outline to appropriate size

+ Grid-fitting the scaled outline to a grid of pixels follow the associated instructions (where are instructions stored?)

+ Create a bitmap image from grid-fitted outline for raster display

=====================================================

1. Scaling the master outline:

+ Get a master outline description for each glyph by requesting a particular glyph at a specific size -> font engine will create the necessary bitmap.

+ Scaler responsible for scaling the master outline to the desired size.

**Converting FUnits to pixels**

The scaler converts values in the master coordinate system to values in the pixel coordinate system by multiplying them by a scale. This scale is:

pointSize \* resolution / (72 points per inch \* units\_per\_em).

where *pointSize* is the size at which the glyph is to be displayed, *resolution* is the resolution of the output device and *units per em* is the resolution of the grid of which the master outline was originally defined. The 72 in the denominator represents the number of points per inch.

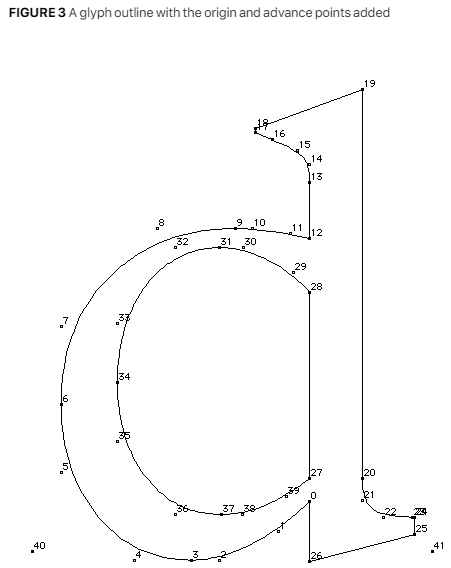
For example, assume that a glyph feature is 550 FUnits in length and defined on a master grid with 2048 units per em. The following calculation reveals that its size on a 72 dpi screen at 18 points, is 4.83 pixels.

550 \* [(18 \* 72 )/ (72 \* 2048 )] = 4.83

**Creating the origin point and the advance point**

The scaler creates two additional points using the data in the ['hdmx'](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM06/Chap6hdmx.html) table in the font file.

The origin point and the advance point are accessible to instructions. If the points in a given glyph are numbered from 0 to n-1, the origin point would have the number n and the advance point the number n+1.



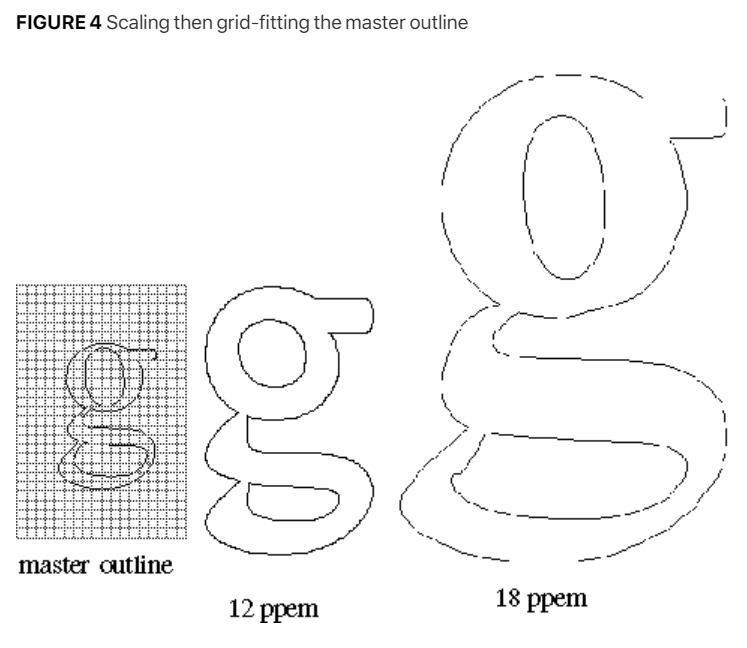
* The origin point is 40 and the advanced point is 41.

**Grid-fitting a scaled outline**

* Bring the points of scaled outline to new location in pixels grid. Their coordinates are indicated as (x,y) positions in the coordinate grid.

Once the master outline for a particular glyph has been scaled to the desired size and device, the instructions associated with that glyph can be executed.

FIGURE 4 shows a master outline and two grid-fitted outlines produced from that master by executing the associated glyph instructions. The grid-fitted outline produced by first scaling the master outline to 12 pixels per em and then executing the glyph instructions differs from that produced by first scaling the master outline to 18 pixels per em and then executing the glyph instructions.



**Scan converting a grid-fitted outline**

Once the master outline has been scaled and grid-fitted, it is ready to be rasterized by the **scan converter**. The scan converter takes the grid-fitted outline and applies a set of rules to determine which pixels will be part of the glyph image when printed or displayed on the screen.

*\* Rule 1: If a pixel's center falls within or on the glyph outline, that pixel is turned on and becomes part of the bitmap image of the glyph.*

The TrueType scan converter solves this problem by using the non-zero winding number rule to distinguish the interior from the exterior of a glyph. This rule is as follows:

“*Points that have a non-zero winding number are inside the glyph. All other points are outside the glyph.”*

*a glyph contour crosses the ray from right to left or bottom to top -> on-transition (plus one).*

*a contour of the glyph crosses the ray from left to right or top to bottom -> off-transition (subtract one).*

* Non-zero winding numbers: interior point
* Zero winding number: exterior point

The direction of a contour can be determined by looking at the point numbers that define the contour. Its direction is from a lower point number toward a higher point number.

Changing the value of scan control is a task accomplished using the **SCANCTRL[]** instruction. See [SCANCTRL[] SCAN conversion ConTRoL.](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SCANCTRL) for more on changing the value of scan control.

\* *Rule 2a: If a horizontal scan line connecting two adjacent pixel centers is intersected by both an on-transition contour and an off-transition contour, and neither of the two pixels was already turned on by rule 1, turn on the left-most pixel.*

\* *Rule 2b: If a vertical scan line connecting two adjacent pixel centers is intersected by both an on-transition contour and an off-transition contour, and neither of the two pixels was already turned on by rule 1, turn on the bottom-most pixel.*

*\* Rule 3a: If a horizontal scan line connecting two adjacent pixel centers is intersected by both an on-transition contour and an off-transition contour, neither of the pixels was already turned on by rule 1, and the two contours continue on to intersect other scan lines (this is not a 'stub'), turn on the left-most pixel.*

*\* Rule 3b: If a vertical scan line connecting two adjacent pixel centers is intersected by both an on-transition contour and an off-transition contour, neither of the pixels was already turned on by rule 1, and the two contours continue on to intersect other scan lines (this is not a 'stub'),turn on the bottom-most pixel.*

**The interpreter environment**

The interpreter is the portion of the TrueType interpreter that executes the instructions found in a font file.

**Where instructions can be used**

Instructions can be associated with particular glyphs or can be associated with a font as a whole.

Instructions associated with a particular glyph are termed a ***glyph program***. Instructions can also be used in the ***font program*** and the ***control value program***.

The *font program* (found in the **'fpgm'** table in the font file) is a set of instructions executed once, the first time a font is accessed by an application. Functions and instructions defined in the *font program* can be accessed in the individual *glyph programs.*

The *control value program* is a sequence of instructions executed every time the point size or transformation changes. These instructions are stored in the **'prep'** table of the font file.

Instructions that belong to *glyph programs* are stored in the **'glyf'** table of the font file. Instructions associated with a glyph are executed every time that glyph is requested.

**Instruction names**

Instructions are uniquely specified by their **opcodes** but are more commonly referred to by their **names**.

Instruction names are of the form MNEMONIC[flag] where the mnemonic is intended as an aid to remembering the instruction's function. For example, the **MDAP** in MDAP[a] instruction stands for **Move Direct Absolute Point**. Similarly, **RUTG[ ]** is short for **Round Up To Grid**.

To calculate the **opcode** for an **instruction variant**, add the unsigned binary number represented by the flag to the lower of the two opcode values given in the documentation. In performing this operation, note that the left most bit is the most significant.

The flags that follow an instruction name also serve to define the semantic meaning of each instruction variant. The binary number is decomposed into a sequence of flags. Flags set to 1 represent TRUE. Flags set to 0 represent FALSE.

**The graphics state**

The graphics state consists of a set of variables that guide the actions of the interpreter. A complete list of the graphics state variables can be found in ["The Graphics State"](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM04/Chap4.html).

The instructions that set the value of a graphics state variable are listed in Table 1.

**Table 1** Setting graphics state values

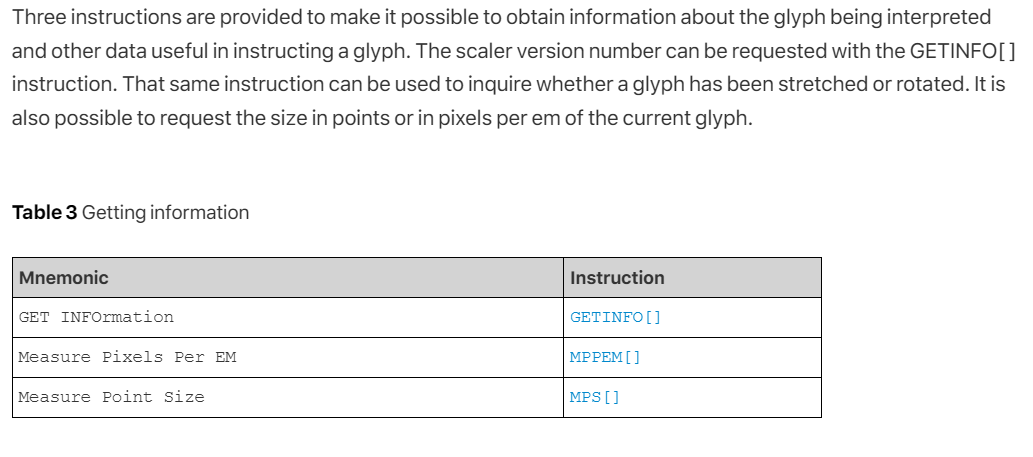
|  |  |  |  |
| --- | --- | --- | --- |
| **Graphics state variable** | **Mnemonic** | **Instruction** |  |
| freedom & projection vector | Set Vectors To Coordinate Axis | [SVTCA[a]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SVTCA) |  | |
| projection vector | Set Projection Vector To Coordinate Axis | [SPVTCA[a]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SPVTCA) |  | |
| freedom vector | Set Freedom Vector To Coordinate Axis | [SFVTCA[a]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SFVTCA) |  | |
| projection vector | Set Projection Vector To Line | [SPVTL[a]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SPVTL) |  | |
| freedom vector | Set Freedom Vector To Line | [SFVTL[a]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SFVTL) |  | |
| freedom vector | Set Freedom Vector To Projection Vector | [SFVTPV[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SFVTPV) |  | |
| dual projection vector | Set Dual Projection Vector To Line | [SDPVTL[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SDPVTL) |  | |
| projection vector | Set Projection Vector To Line | [SVPTL[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SPVTL) |  | |
| projection vector | Set Projection Vector From Stack | [SPVFS[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SPVFS) |  | |
| freedom vector | Set Freedom Vector From Stack | [SFVFS[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SFVFS) |  | |
| rp0 | Set Reference Point 0 | [SRP0[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SRPO) |  | |
| rp1 | Set Reference Point 1 | [SRP1[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SRP1) |  | |
| rp2 | Set Reference Point 2 | [SRP2[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SRP2) |  | |
| zp0 | Set Zone Pointer 0 | [SZP0[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SZP0) |  | |
| zp1 | Set Zone Pointer 1 | [SZP1[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SZP1) |  | |
| zp2 | Set Zone Pointer 2 | [SZP2[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SZP2) |  | |
| zp0, zp1, zp2 | Set Zone PointerS | [SZPS[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SZPS) |  | |
| round state | Round To Half Grid | [RTHG[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#RTHG) |  | |
| round state | Round To Grid | [RTG[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#RTG) |  | |
| round state | Round To Double Grid | [RTDG[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#RTDG) |  | |
| round state | Round Up To Grid | [RUTG[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#RUTG) |  | |
| round state | Round Down To Grid | [RDTG[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#RDTG) |  | |
| round state | set Rounding Off | [ROFF[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#ROFF) |  | |
| round state | Super ROUND | [SROUND[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SROUND) |  | |
| round state | Super 45 ROUND | [S45ROUND[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#S45ROUND) |  | |
| loop | Set LOOP | [SLOOP[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SLOOP) |  | |
| single width cut-in | Set Single Width Cut-In | [SSWCI[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SSWCI) |  | |
| control value cut-in | Set Control Value Table Cut-In | [SCVTCI[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SCVTCI) |  | |
| minimum distance | Set Minimum Distance | [SMD[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#SMD) |  | |

Table 2 lists the two get instructions.

**Table 2** : Getting graphics state variables

|  |  |
| --- | --- |
| **Mnemonic** | **Instruction** |
| Get Freedom Vector | [GFV[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#GFV) |
| Get Projection Vector | [GPV[]](https://developer.apple.com/fonts/TrueType-Reference-Manual/RM05/Chap5.html#GPV) |

**Getting information from the interpreter**



**Managing points in the pixel grid**

The key task of the TrueType instruction set is one of grid-fitting glyph outlines to allow the scan converter to produce superior bitmap images for display. That task consists of reshaping glyph outlines by moving the points that make up their outline. The following sections describe the instructions that are used to manage points in the pixel grid.

**Zones and points**

Points are locations in a grid. As stored in the font file, points have coordinates that are expressed in FUnits. These coordinates refer to positions in the master grid. Once scaled by the font engine, however, point locations are expressed as 26.6 fixed point numbers representing locations in the device specific pixel grid. That is, they are given to the nearest sixty-fourth of a pixel. The notation wn:fp is used to express point locations where wn refers to a whole number and fp refers to the fractional part. The number six would be written 6:0. The number one fourth would be written 0:16. When it is convenient, point locations will be expressed as decimal numbers, such as 2.5 to represent the position two and one-half.

**Movement in the pixel grid**

Movement of points in the pixel grid is always along the *freedom vector*, a graphics state variable representing a vector in the pixel grid. The *freedom vector* can be translated in space so that it maintains its orientation relative to the coordinate system but passes through the point that is to be moved. By convention, the *freedom vector* is shown as a dotted line passing through a point that is to be moved. Points move along the vector.

**Getting and setting the freedom vector**

One instruction exists for obtaining the current value of the freedom vector. Its value is returned as a pair of (x,y) coordinates. The freedom vector can be set by specifying its coordinates on the stack, by setting it to either or the coordinate axes, by setting it to be parallel or perpendicular to a line or by setting it to be parallel or perpendicular to the projection vector. The instructions for setting the freedom vector are listed in Table 15.

**Setting and getting point locations**