**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 3**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 3 m** **56342**

**Online – April 2021**

**Title: Adding support for non-variable paint table formats in ISO/IEC 14496-22 AMD2**

**Authors: Peter Constable, Cosimo Lupo, Dominik Röttsches, Roderick Sheeter, Vladimir Levantovsky (on behalf of AHG)**

In the current description of the COLR version 1 table included as part of the Working Draft amendment 2 (output document N0136\_19969), various paint table formats are defined. These table formats include structures that allow using delta-set indices to support variable fonts implementations. However, if color glyphs are designed as static instances, these fields are still present, but unused, imposing a significant size penalty without any benefit for non-variable fonts.

For example, the size of paint table specifying a linear gradient is 40 bytes, and 24 (60%) of those bytes are allocated for delta-set indices that would remain unused in non-variable fonts. Many other paint table formats would have ~50% redundancy for the same reasons. While the actual size impact of these redundancies on a non-variable font would depend on the actual color glyph descriptions, eliminating the un-used delta-set indices in various paint table formats would result in significant font size reduction; e.g. doing so for “Noto Emoji” font would allow reducing its file size by 16%.

Variations can also be applied to transforms; therefore, both variable and non-variable transform formats can be defined. However, if color glyph descriptions include variable transformation, they may cause portions of a color glyph to go outside the bounding box of the default glyph and need to be accounted for.

This contribution proposes new updates to enable support for both variable and non-variable paint table formats as part of the COLR version 1 table format defined by the Working Draft of ISO/IEC 14496-22:2019/AMD2.

*5.7.11.1 “Graphic compositions”*

*Replace the fifth paragraph (immediately below the Figure 5.8) with the following:*

The basic concepts also apply to color glyphs defined using the version 1 formats: shapes have fills and can be arranged in layers. But the additional formats of version 1 support much richer capabilities. In a version 1 color glyph, graphic constructs and capabilities are represented primarily in *Paint* tables, which are linked together in a *directed, acyclic graph*. Several different Paint formats are defined, each describing a particular type of graphic operation:

* A PaintColrLayers table provides a layering structure used for creating a color glyph from layered elements. A PaintColrLayers table can be used at the root of the graph, providing a base layering structure for the entire color glyph definition. A PaintColrLayers table can also be nested within the graph, providing a set of layers to define some graphic sub-component within the color glyph.
* The PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, and PaintVarSweepGradient tables provide basic fills, using color entries from the CPAL table.
* The PaintGlyph table provides glyph outlines as the basic shapes.
* The PaintTransform and PaintVarTransform tables are used to apply an affine transformation matrix to a sub-graph of paint tables, and the graphic operations they represent. The PaintTranslate, PaintVarTranslate, PaintRotate, PaintVarRotate, PaintSkew, and PaintVarSkew tables support specific transformations.
* The PaintComposite table supports alternate compositing and blending modes for two sub-graphs.
* The PaintColrGlyph table allows a color glyph definition, referenced by a base glyph ID, to be re-used as a sub-graph within multiple color glyphs.

NOTE Some paint formats come in *Paint\** and *PaintVar\** pairs. In these cases, the latter format supports variations in variable fonts, while the former provides a more compact representation for the same graphic capability but without variation capability.

*5.7.11.1.1 “Colors and solid color fills”*

*Replace the second and third paragraphs with the following:*

The CPAL color data includes alpha information, as well as RGB values. In the COLR version 0 formats, a color reference is made in LayerRecord as a palette entry index alone. In the formats added for COLR version 1, a color reference is made in a color index record, which includes a palette entry index and a separate alpha value. Separation of alpha from palette entries in version 1 allows use of transparency in a color glyph definition independent of the choice of palette. The alpha value in the color index record is multiplied into the alpha value given in the CPAL color entry.

Two color index record formats are defined: ColorIndex, and VarColorIndex. The latter can be used in variable fonts to make the alpha value variable.

In version 1, a solid color fill is specified using a PaintVarSolid or PaintSolid table, with or without variation support, respectively. See 5.7.11.2.5.2 for format details.

See 5.7.11.1.3 for details on how fills are applied to a shape.

*5.7.11.1.2.1 “Colors lines”*

*In the very end of the sub-clause, add the following two paragraphs:*

Color lines are specified using color line tables, which contain arrays of color stop records. Two color line table and two color stop record formats are defined:

* ColorLine table and ColorStop record
* VarColorLine table and VarColorStop record

The VarColorLine and VarColorStop formats can be used in variable fonts and allow for stop offsets to be variable. The VarColorStop record also uses the VarColorIndex record, allowing the alpha to be variable. The ColorLine and ColorStop formats provide a more compact representation when variation is not required. See 5.7.11.2.4 for format details.

*5.7.11.1.3 “Filling shapes”*

*Replace the third paragraph (immediately preceding the Figure 5.33) with the following:*

Any of the basic fill formats (PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, PaintVarSweepGradient) can be used as the child paint table. This is illustrated in figure 5.33: a PaintGlyph table has a glyph ID for an outline in the shape of a triangle, and it links to a child PaintLinearGradient table. The combination is used to represent a triangle filled with the linear gradient.

*5.7.11.1.3 “Filling shapes”*

*Replace the fourth and fifth paragraphs (immediately below the Figure 5.33) with the following:*

The child of a PaintGlyph table is not, however, limited to one of the basic fill formats. Rather, the child can be the root of a sub-graph that describes some graphic composition that is used as a fill. Another way to describe the relationship between a PaintGlyph table and its child sub-graph is that the glyph outline specified by the PaintGlyph table defines a bounds, or *clip region*, that is applied to the fill composition defined by the child sub-graph.

To illustrate this, the example in figure 5.33 is extended in figure 5.34 so that a PaintGlyph table links to a second PaintGlyph that links to a PaintLinearGradient: the parent PaintGlyph will clip the filled shape described by the child sub-graph.

*5.7.11.1.4 “Layering”*

*Replace the fifth paragraph (immediately below the Figure 5.33) with the following:*

The version 1 formats define a color glyph as a directed, acyclic graph of paint tables, and the concept of layering corresponds roughly to the number of distinct leaf nodes in the graph. (See 5.7.11.1.9.) The basic fill formats (PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, PaintVarSweepGradient) do not have child paint tables and so can only be leaf nodes in the graph. Some paint tables, such as the PaintGlyph table, have only a single child, so can be used within a layer but do not provide any means of adding additional layers. Increasing the number of layers requires paint tables that have two or more children, creating a fork in the graph.

*5.7.11.1.5 “Transformations”*

*Replace the first two paragraphs (immediately before the Figure 5.39) with the following:*

A 2 × 3 transformation matrix can be used within a color glyph description to apply an affine transformation to a sub-graph in the color glyph description. Affine transformations supported by a matrix can be a combination of scale, skew, mirror, rotate, or translate. The transformation is applied to all nested paints in the child sub-graph.

A transformation matrix is specified using a PaintVarTransform or PaintTransform table, with or without variation support, respectively. See 5.7.11.2.5.8 for format details.

The effect of a transformation is illustrated in figure 5.39: a PaintTransform table is used to specify a rotation, and both the glyph outline and gradient in the sub-graph are rotated.

*5.7.11.1.5 “Transformations”*

*Replace the third paragraph (immediately before the Figure 5.40) with the following:*

If the sub-graph of a transformation table contains another nested transformation table, then the second transformation also applies to its child sub-graph. For the sub-sub-graph, the two transformations are combined. To illustrate this, the example in figure 5.39 is extended in figure 5.40 by inserting a mirroring transformation between the PaintGlyph and PaintLinearGradient tables: the glyph outline is rotated as before, but the gradient is mirrored in its (pre-rotation) y-axis as well as being rotated. Notice that both visible elements—the shape and the gradient fill—are affected by the rotation, but only the gradient is affected by the mirroring.

*5.7.11.1.5 “Transformations”*

*Replace the fourth and fifth paragraphs (immediately below the Figure 5.40, including the NOTE) with the following:*

While the PaintTransform and PaintVarTransform tables support several types of transforms, addition paint formats are defined to support specific transformations:

* PaintVarTranslate and PaintTranslate support translation only, with or without variation support, respectively. See 5.7.11.2.5.9 for format details.
* PaintVarRotate and PaintRotate support rotation only, with or without variation support, respectively. See 5.7.11.2.5.10 for format details.
* PaintVarSkew and PaintSkew support skew only, with or without variation support, respectively. See 5.7.11.2.5.11 for format details.

When only one of these specific types of transformation is required, these formats provide a more compact representation than the PaintTransform or PaintVarTransform formats. Another significant difference of the rotation and skew formats is that the rotations and skews are specified as angles, in counter-clockwise degrees.

NOTE Specifying the rotation or skew as an angle can have a significant benefit in variable fonts if an angle of skew or rotation needs to vary, since it is easier to implement variation of angles when specified directly rather than as matrix elements. This is because the matrix elements for a rotation or skew are the sine, cosine or tangent of the rotation angle, which do not change in linear proportion to the angle. To achieve a linear variation of rotation using matrix elements would require approximating the variation using multiple delta sets.

The rotations and skews specified using PaintRotate, PaintVarRotate, PaintSkew, or PaintVarSkew tables can also be representated as a matrix using a PaintTransform or PaintVarTransform table. If a PaintRotate, PaintVarRotate, PaintSkew, or PaintVarSkew table is used in combination with a PaintTransform or PaintVarTransform table, the combined behavior shall be the same as if the rotation or skew were represented using an equivalent matrix. See 5.7.11.2.5.10 for details regarding the matrix equivalent for a rotation expressed as an angle; and see 5.7.11.2.5.11 for similar details in relation to skews.

*5.7.11.1.7.1 “Re-use by referencing shared subtables”*

*Replace the first paragraph with the following:*

Several of the paint table formats link to a child paint table using a forward offset within the file:

* PaintGlyph
* PaintComposite
* PaintTransform, PaintVarTransform
* PaintTranslate, PaintVarTranslate
* PaintRotate, PaintVarRotate
* PaintSkew, PaintVarSkew

A child subtable can be shared by several tables of these formats. For example, several PaintGlyph tables might link to the same PaintSolid table, or to the same node for a sub-graph describing a more complex fill. The only limitation is that child paint tables are referenced using a forward offset from the start of the referencing table, so a re-used paint table can only occur later in the file than any of the paint tables that use it.

*5.7.11.1.8.2 “Metrics and boundedness of color glyphs using version 1 formats”*

*Replace the third and fourth (the last two) paragraphs with the following:*

A valid color glyph definition shall define a bounded region—that is, it shall paint within a region for which a finite bounding box could be defined. The different paint formats have different boundedness characteristics:

* PaintGlyph is inherently bounded.
* PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, and PaintVarSweepGradient are inherently unbounded.
* PaintColrLayers is bounded *if and only if* all referenced sub-graphs are bounded.
* PaintColrGlyph is bounded *if and only if* the color glyph definition for the referenced base glyph ID is bounded.
* PaintTransform, PaintVarTransform, PaintTranslate, PaintVarTranslate, PaintRotate, PaintVarRotate, PaintSkew, and PaintVarSkew are bounded *if and only if* the referenced sub-graph is bounded.
* PaintComposite is either bounded or unbounded, according to the composite mode used and the boundedness of the referenced sub-graphs. See 5.7.11.2.5.12 for details.

Applications shall confirm that a color glyph definition is bounded, and shall not render a color glyph if the defining graph is not bounded.

To ensure that rendering implementations do not clip any part of a color glyph, the bounding box of the base glyph needs to be large enough to encompass the entire color glyph composition. In a variable font, glyph outlines can vary, but transformations in a color glyph description can also vary, affecting the portions of the design grid to be painted. For example, a filled rectangle that is wide but not tall for one variation instance can be variably rotated to be tall but not wide for other instances. The bounding box of the base glyph either should be large enough to encompass the color glyph for all instances, or should itself vary such that each instance bounding box encompasses the instance color glyph.

*5.7.11.1.9 “Color glyphs as a directed acyclic graph”*

*Replace the third paragraph with the following:*

The graph for a color glyph is a combination of paint tables using any of the paint table formats. The simplest color glyph definition would consist of a PaintGlyph table linked to a basic fill table (PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, PaintVarSweepGradient). But the graph can be arbitrarily complex, with an arbitrary depth of paint nodes (to the limits inherent in the formats).

*5.7.11.1.9 “Color glyphs as a directed acyclic graph”*

*Replace the NOTE after the sixth paragraph with the following:*

NOTE These constraints imply that all leaf nodes will be one of PaintSolid, PaintVarSolid, PaintLinearGradient, PaintVarLinearGradient, PaintRadialGradient, PaintVarRadialGradient, PaintSweepGradient, or PaintVarSweepGradient.

*5.7.11.2.4 “ColorIndex, ColorStop and ColorLine”*

*Replace the first paragraph and the ColorIndex record table with the following:*

Colors are used in solid color fills for graphic elements, or as *stops* in a color line used to define a gradient. Colors are defined by reference to palette entries in the CPAL table (5.7.12). While CPAL entries include an alpha component, color-index records for referencing palette entries are defined here that includes a separate alpha specification to allow different graphic elements to use the same color but with different alpha values, and to allow for variation of the alpha in variable fonts.

Two color-index record formats are defined: one that allows for variation of alpha, and one that does not.

*ColorIndex record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | paletteIndex | Index for a CPAL palette entry. |
| F2DOT14 | alpha | Alpha value. |

*VarColorIndex record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | paletteIndex | Index for a CPAL palette entry. |
| VarF2Dot14 | alpha | Variable alpha value. |

*5.7.11.2.4 “ColorIndex, ColorStop and ColorLine”*

*Replace the ColorStop record table with the following:*

Two color-stop record formats are defined: one that allows for variation of stop offset position or of alpha, and one that does not.

*ColorStop record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| F2DOT14 | stopOffset | Position on a color line. |
| ColorIndex | color |  |

*VarColorStop record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarF2Dot14 | stopOffset | Position on a color line; variable. |
| VarColorIndex | color |  |

*5.7.11.2.4 “ColorIndex, ColorStop and ColorLine”*

*Replace the ColorLine record table with the following:*

Two color-line table formats are defined: one that allows for variation of color stop offsets positions or of alpha values, and one that does not. Different paint table formats for gradients use one or the other of the color line formats.

*ColorLine table:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | extend | An Extend enum value. |
| uint16 | numStops | Number of ColorStop records. |
| ColorStop | colorStops[numStops] |  |

*VarColorLine table:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | extend | An Extend enum value. |
| uint16 | numStops | Number of ColorStop records. |
| VarColorStop | colorStops[numStops] | Allows for variations. |

*5.7.11.2.5 “Paint tables”*

*Replace the first two paragraphs with the following:*

Paint tables are used for COLR version 1 color glyph definitions. Twenty Paint table formats are defined (formats 1 to 20). Some formats come in non-variable and variable pairs, but otherwise, each provides different graphic capability for defining the composition for a color glyph. The graphic capability of each format and the manner in which they are combined to represent a color glyph has been described above—see 5.7.11.1.

Each paint table format has a one-byte format field as the first field. When parsing font data, the format field can be read first to determine the format of the table.

*5.7.11.2.5.2 “Format 2: PaintSolid”*

*Rename the subclause to “Formats 2 and 3: PaintSolid, PaintVarSolid” and replace the content of the entire subclause with the following:*

Formats 2 and 3 are used to specify a solid color fill. Format 3 allows for variation of alpha in a variable font; format 2 provides a more compact representation when variation is not required. Format 3 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information about specifying color values, see 5.7.11.1.1. For information about applying a fill to a shape, see 5.7.11.1.3.

*PaintSolid table (format 2):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 2. |
| ColorIndex | color | ColorIndex record for the solid color fill. |

*PaintVarSolid table (format 3):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 3. |
| VarColorIndex | color | VarColorIndex record for the solid color fill. |

For the ColorIndex and VarColorIndex record formats, see 5.7.11.2.4.

*5.7.11.2.5.3 “Format 3: PaintLinearGradient”*

*Rename the subclause to “Formats 4 and 5: PaintLinearGradient, PaintVarLinearGradient” and replace the content of the entire subclause with the following:*

Formats 4 and 5 are used to specify a linear gradient fill. Format 4 allows for variation of color stop positions or of alpha in a variable font; format 5 provides a more compact representation when variation is not required. Format 5 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information about linear gradients, see 5.7.11.1.2.2. For information about applying a fill to a shape, see 5.7.11.1.3.

The PaintLinearGradient and PaintVarLinearGradient tables have a ColorLine and VarColorLine subtable, respectively. For the ColorLine and VarColorLine table formats, see 5.7.11.2.4. For background information on the color line, see 5.7.11.1.2.1.

*PaintLinearGradient table (format 4):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 4. |
| Offset24 | colorLineOffset | Offset to ColorLine table. |
| FWORD | x0 | Start point (p₀) x coordinate. |
| FWORD | y0 | Start point (p₀) y coordinate. |
| FWORD | x1 | End point (p₁) x coordinate. |
| FWORD | y1 | End point (p₁) y coordinate. |
| FWORD | x2 | Rotation point (p₂) x coordinate. |
| FWORD | y2 | Rotation point (p₂) y coordinate. |

*PaintVarLinearGradient table (format 5):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 5. |
| Offset24 | colorLineOffset | Offset to VarColorLine table. |
| VarFWord | x0 | Start point (p₀) x coordinate. |
| VarFWord | y0 | Start point (p₀) y coordinate. |
| VarFWord | x1 | End point (p₁) x coordinate. |
| VarFWord | y1 | End point (p₁) y coordinate. |
| VarFWord | x2 | Rotation point (p₂) x coordinate. |
| VarFWord | y2 | Rotation point (p₂) y coordinate. |

*5.7.11.2.5.4 “Format 4: PaintRadialGradient”*

*Rename the subclause to “Formats 6 and 7: PaintRadialGradient, PaintVarRadialGradient” and replace the content of the entire subclause with the following:*

Format 6 and 7 are used to specify a radial gradient fill. Format 7 allows for variation of color stop positions or of alpha in a variable font; format 6 provides a more compact representation when variation is not required. Format 7 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information about radial gradients supported in COLR version 1, see 5.7.11.1.2.3. For information about applying a fill to a shape, see 5.7.11.1.3.

The PaintRadialGradient and PaintVarRadialGradient tables have a ColorLine and VarColorLine subtable, respectively. For the ColorLine and VarColorLine table formats, see in 5.7.11.2.4. For background information on the color line, see 5.7.11.1.2.1.

*PaintRadialGradient table (format 6):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 6. |
| Offset24 | colorLineOffset | Offset to ColorLine table. |
| FWORD | x0 | Start circle center x coordinate. |
| FWORD | y0 | Start circle center y coordinate. |
| UFWORD | radius0 | Start circle radius. |
| FWORD | x1 | End circle center x coordinate. |
| FWORD | y1 | End circle center y coordinate. |
| UFWORD | radius1 | End circle radius. |

*PaintVarRadialGradient table (format 7):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 7. |
| Offset24 | colorLineOffset | Offset to VarColorLine table. |
| VarFWord | x0 | Start circle center x coordinate. |
| VarFWord | y0 | Start circle center y coordinate. |
| VarUFWord | radius0 | Start circle radius. |
| VarFWord | x1 | End circle center x coordinate. |
| VarFWord | y1 | End circle center y coordinate. |
| VarUFWord | radius1 | End circle radius. |

*5.7.11.2.5.5 “Format 5: PaintGlyph”*

*Rename the subclause to “Formats 8 and 9: PaintSweepGradient, PaintVarSweepGradient” and replace the content of the entire subclause with the following:*

Format 8 and 9 are used to specify a sweep gradient fill. Format 9 allows for variation of color stop positions or of alpha in a variable font; format 8 provides a more compact representation when variation is not required. Format 9 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information about sweep gradients, see 5.7.11.1.2.4. For information about applying a fill to a shape, see 5.7.11.1.3.

The PaintSweepGradient and PaintVarSweepGradient table have a ColorLine and VarColorLine subtable, respectively. For the ColorLine and VarColorLine table formats, see 5.7.11.2.4. For background information on the color line, see 5.7.11.1.2.1.

*PaintSweepGradient table (format 8):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 8. |
| Offset24 | colorLineOffset | Offset to ColorLine table. |
| FWORD | centerX | Center x coordinate. |
| FWORD | centerY | Center y coordinate. |
| Fixed | startAngle | Start of the angular range of the gradient. |
| Fixed | endAngle | End of the angular range of the gradient. |

*PaintVarSweepGradient table (format 9):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 9. |
| Offset24 | colorLineOffset | Offset to VarColorLine table. |
| VarFWord | centerX | Center x coordinate. |
| VarFWord | centerY | Center y coordinate. |
| VarFixed | startAngle | Start of the angular range of the gradient. |
| VarFixed | endAngle | End of the angular range of the gradient. |

*Insert new subclause 5.7.11.2.5.6 “Format 10: PaintGlyph” with the following content, and re-number the remaining sub-clauses.*

Format 10 is used to specify a glyph outline to use as a shape to be filled or, equivalently, a clip region. The outline sets a clip region that constrains the content of a separate paint subtable and the sub-graph linked from that subtable.

For information about applying a fill to a shape, see 5.7.11.1.3.

*PaintGlyph table (format 10):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 10. |
| Offset24 | paintOffset | Offset to a Paint table. |
| uint16 | glyphID | Glyph ID for the source outline. |

The glyphID value shall be less than the numGlyphs value in the ‘maxp’ table (5.2.6). That is, it shall be a valid glyph with outline data in the ‘glyf’ (5.3.4), ‘CFF’ (5.4.2) or CFF2 (5.4.3) table. Only that outline data is used. In particular, if this glyph ID has a description in the COLR table (glyphID appears in a COLR BaseGlyph record or the BaseGlyphV1List), that COLR data is not relevant for purposes of the PaintGlyph table.

*5.7.11.2.5.7*

*Rename the newly renumbered subclause 5.7.11.2.5.7 “Format 6: PaintColrGlyph” to “Format 11: PaintColrGlyph”, and update format number to “11” in the first paragraph and in both PaintColorGlyph table header and format field description.*

*5.7.11.2.5.8*

*Rename the newly renumbered subclause 5.7.11.2.5.8 “Format 7: PaintTransformed” to “Formats 12 and 13: PaintTransform. PaintVarTransform”, and replace the content of the entire subclause with the following:*

Formats 12 and 13 are used to apply an affine transformation to a sub-graph. The paint table that is the root of the sub-graph is linked as a child.

Format 13 allows for variation of the transformation in a variable font; format 12 provides a more compact representation when variation is not required. Format 13 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information regarding transformations in a color glyph definition, see 5.7.11.1.5.

*PaintTransform table (format 12):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 12. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| Affine2x3 | transform | An Affine2x3 record (inline). |

*PaintVarTransform table (format 13):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 13. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| VarAffine2x3 | transform | A VarAffine2x3 record (inline). |

The affine transformation is defined by a 2×3 matrix, specified in an Affine2x3 or VarAffine2x3 record. The 2×3 matrix supports scale, skew, reflection, rotation, and translation transformations. The matrix elements in the VarAffine2x3 record use VarFixed records, allowing the transform definition to be variable in a variable font.

*Affine2x3 record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Fixed | xx | x-component of transformed x-basis vector |
| Fixed | yx | y-component of transformed x-basis vector |
| Fixed | xy | x-component of transformed y-basis vector |
| Fixed | yy | y-component of transformed y-basis vector |
| Fixed | dx | Translation in x direction. |
| Fixed | dy | Translation in y direction. |

*VarAffine2x3 record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarFixed | xx | x-component of transformed x-basis vector |
| VarFixed | yx | y-component of transformed x-basis vector |
| VarFixed | xy | x-component of transformed y-basis vector |
| VarFixed | yy | y-component of transformed y-basis vector |
| VarFixed | dx | Translation in x direction. |
| VarFixed | dy | Translation in y direction. |

For a pre-transformation position *(x, y)*, the post-transformation position *(x′, y′)* is calculated as follows:

*x′* = *xx* \* *x* + *xy* \* *y* + *dx*  
*y′* = *yx* \* *x* + *yy* \* *y* + *dy*

NOTE It is helpful to understand linear transformations by their effect on *x-* and *y-basis* vectors *î* = (1, 0) and *ĵ* = (0, 1). The transform described by the Affine2x3 or VarAffine2x3 record maps the basis vectors to *î′* = (*xx*, *yx*) and *ĵ′* = (*xy*, *yy*), and translates the origin to (*dx, dy*).

When the transformed composition from the referenced paint table (and its sub-graph) is composed into the destination (represented by the parent of this table), the source design grid origin is aligned to the destination design grid origin. The transform can translate the source such that a pre-transform position (0,0) is moved elsewhere. The *post-transform* origin, (0,0), is aligned to the destination origin.

*5.7.11.2.5.9*

*Rename the newly renumbered subclause 5.7.11.2.5.9 “Format 8: PaintTranslate” to “Formats 14 and 15: PaintTranslate. PaintVarTranslate”, and replace the content of the entire subclause with the following:*

Formats 14 and 15 are used to apply a translation to a sub-graph. The paint table that is the root of the sub-graph is linked as a child.

Format 15 allows for variation of the translation in a variable font; format 14 provides a more compact representation when variation is not required. Format 15 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information regarding transformations in a color glyph definition, see 5.7.11.1.5.

*PaintTranslate table (format 14):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 14. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| Fixed | dx | Translation in x direction. |
| Fixed | dy | Translation in y direction. |

*PaintVarTranslate table (format 15):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 15. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| VarFixed | dx | Translation in x direction. |
| VarFixed | dy | Translation in y direction. |

NOTE Pure translation can also be represented using the PaintTransform or PaintVarTransform table by setting *xx* = 1, *yy* = 1, *xy* and *yx* = 0, and setting *dx* and *dy* to the translation values. The PaintTranslate or PaintVarTranslate table provides a more compact representation when only translation is required.

The translation will result in the pre-transform position (0,0) being moved elsewhere. See 5.7.11.2.5.8 regarding alignment of the transformed content with the destination.

*5.7.11.2.5.10*

*Rename the newly renumbered subclause 5.7.11.2.5.10 “Format 9: PaintRotate” to “Formats 16 and 17: PaintRotate. PaintVarRotate”, and replace the content of the entire subclause with the following:*

Formats 16 and 17 are used to apply a rotation to a sub-graph. The paint table that is the root of the sub-graph is linked as a child. The amount of rotation is expressed directly as an angle, and X and Y coordinates can be provided for the center of rotation.

Format 17 allows for variation of the rotation in a variable font; format 16 provides a more compact representation when variation is not required. Format 17 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information regarding transformations in a color glyph definition, see 5.7.11.1.5.

*PaintRotate table (format 16):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 16. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| Fixed | angle | Rotation angle, in counter-clockwise degrees. |
| Fixed | centerX | x coordinate for the center of rotation. |
| Fixed | centerY | y coordinate for the center of rotation. |

*PaintVarRotate table (format 17):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 17. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| VarFixed | angle | Rotation angle, in counter-clockwise degrees. |
| VarFixed | centerX | x coordinate for the center of rotation. |
| VarFixed | centerY | y coordinate for the center of rotation. |

NOTE Pure rotation about a point can also be represented using the PaintTransform or PaintVarTransform table. For rotation about the origin, this could be done by setting matrix values as follows for angle θ:

* *xx* = cos(θ)
* *yx* = sin(θ)
* *xy* = -sin(θ)
* *yy* = cos(θ)
* *dx* = *dy* = 0

The important difference of the PaintRotate and PaintVarRotate tables is in allowing an angle to be specified directly in degrees, rather than as changes to basis vectors. In variable fonts, if a rotation angle needs to vary, it is easier to get smooth variation if an angle is specified directly than when using trigonometric functions to derive matrix elements.

When combining the transform effect of a PaintRotate or PaintVarRotate table with other transforms, the result shall be the same as if the rotation were represented using an equivalent matrix.

A rotation can result in the pre-transform position (0, 0) being moved elsewhere. See 5.7.11.2.5.8 regarding alignment of the transformed content with the destination.

*5.7.11.2.5.11*

*Rename the newly renumbered subclause 5.7.11.2.5.11 “Format 10: PaintSkew” to “Formats 18 and 19: PaintSkew. PaintVarSkew”, and replace the content of the entire subclause with the following:*

Formats 18 and 19 are used to apply a skew to a sub-graph. The paint table that is the root of the sub-graph is linked as a child. The amount of skew in the X or Y direction is expressed directly as angles, and X and Y coordinates can be provided for the center of rotation.

Format 19 allows for variation of the rotation in a variable font; format 18 provides a more compact representation when variation is not required. Format 19 shall not be used in non-variable fonts or if the COLR table does not have an ItemVariationStore subtable.

For general information regarding transformations in a color glyph definition, see 5.7.11.1.5.

*PaintSkew table (format 18):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 18. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| Fixed | xSkewAngle | Angle of skew in the direction of the x-axis, in counter-clockwise degrees. |
| Fixed | ySkewAngle | Angle of skew in the direction of the y-axis, in counter-clockwise degrees. |
| Fixed | centerX | x coordinate for the center of rotation. |
| Fixed | centerY | y coordinate for the center of rotation. |

*PaintVarSkew table (format 19):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 19. |
| Offset24 | paintOffset | Offset to a Paint subtable. |
| VarFixed | xSkewAngle | Angle of skew in the direction of the x-axis, in counter-clockwise degrees. |
| VarFixed | ySkewAngle | Angle of skew in the direction of the y-axis, in counter-clockwise degrees. |
| VarFixed | centerX | x coordinate for the center of rotation. |
| VarFixed | centerY | y coordinate for the center of rotation. |

NOTE Pure skews about a point can also be represented using the PaintTransform or PaintVarTransform table. For skews about the origin, this could be done by setting matrix values as follows for *x* skew angle φ and *y* skew angle ψ:

* *xx* = *yy* = 1
* *yx* = tan(ψ)
* *xy* = -tan(φ)
* *dx* = *dy* = 0

The important difference of the PaintSkew and PaintVarSkew tables is in being able to specify skew as an angle, rather than as changes to basis vectors. In variable fonts, if a skew angle needs to vary, it is easier to get smooth variation if an angle is specified directly than when using trigonometric functions to derive matrix elements.

When combining the transform effect of a PaintSkew or PaintVarSkew table with other transforms, the result shall be the same as if the skew were represented using an equivalent matrix.

A skew can result in the pre-transform position (0, 0) being moved elsewhere. See 5.7.11.2.5.8 regarding alignment of the transformed content with the destination.

*5.7.11.2.5.12*

*Rename the newly renumbered subclause 5.7.11.2.5.12 “Format 11: PaintComposite” to “Format 20: PaintComposite”, and update format number to “20” in the first paragraph and in both PaintComposite table header and format field description.*

*5.7.11.3 “COLR version 1 rendering algorithm”*

*In the end of the subclause, replace the pseudo-code function with the following:*

// render a paint table and its sub-graph  
function renderPaint(paint)  
  
 if format 1: // PaintColrLayers  
 for each referenced child paint table, in bottom-up z-order:  
 // for ordering, see 5.7.11.1.4, 5.7.11.2.5.1  
 call renderPaint() passing the child paint table  
  
 compose the returned graphic onto the surface using simple  
 alpha blending  
  
 if format 2 or 3: // PaintSolid, PaintVarSolid  
 paint the specified color onto the surface  
  
 if format 4, 5, 6, 7, 8 or 9:  
 // PaintLinearGradient, PaintVarLinearGradient  
 // PaintRadialGradient, PaintVarRadialGradient  
 // PaintSweepGradient, PaintVarSweepGradient  
 paint the gradient onto the surface following the gradient  
 algorithm  
  
 if format 10: // PaintGlyph  
 apply the outline of the referenced glyph to the clip region  
 // take the intersection of clip regions—see 5.7.11.1.3  
  
 call renderPaint() passing the child paint table  
  
 restore the previous clip region  
  
 if format 11: // PaintColrGlyph  
 call renderPaint() passing the paint table referenced by the base  
 glyph ID  
  
 if format 12, 13, 14, 15, 16, 17, 18 or 19:  
 // PaintTransform, PaintVarTransform  
 // PaintTranslate, PaintVarTranslate  
 // PaintRotate, PaintVarRotate  
 // PaintSkew, PaintVarSkew  
 apply the specified transform  
 // compose the transform with the current transform state—see  
 // 5.7.11.1.5  
  
 call renderPaint() passing the child paint table  
  
 restore the previous transform state  
  
 if format 20: // PaintComposite  
  
 // render backdrop sub-graph  
 call renderPaint() passing the backdrop child paint table and save  
 the result  
  
 // render source sub-graph  
 call renderPaint() passing the source child paint table and save  
 the result  
  
 // compose source and backdrop  
 compose the source and backdrop using the specified composite mode  
  
 // compose final result  
 compose the result of the above composition onto the surface using  
 simple alpha blending

*5.7.11.4 “COLR table and OFF Font Variations”*

*In the end of the subclause, replace the 5th, 6th and 7th paragraphs with the following:*

The value field of these records provides the default value for a given item. The remaining fields provide index values for a particular ItemVariationData subtable and DeltaSet record—the two-level organizational hierarchy used within the Item Variation Store.

If the COLR table does not contain an Item Variation Store subtable, the index fields of these records shall be ignored by applications, and should be set to zero. The value field is read directly without any variation calculation.

If the COLR table contains an Item Variation Store subtable, the index fields shall be used to obtain a delta value that is combined with the value of the value field. In this case, the index fields of the VarFWord, VarUFWord, VarF2Dot14 and VarFixed records shall always be set with specific values. The indices are base 0, therefore 0x0000 cannot be used as an ignorable default. To indicate that an item has no variation data, the index fields shall be set to 0xFFFF/0xFFFF. (See 7.2.3.2.)

For variable fonts that use COLR version 1 formats, special considerations apply to the effect of variation on the bounding box. See 5.7.11.1.8.2 for details.