**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 m55371**

**Online – October 2020**

**Title:** Proposed amendments for extending the COLR table format and functionality

**Author:** Roderick Sheeter ([rsheeter@google.com](mailto:rsheeter@google.com))

**Author:** Dominik Röttsches ([drott@google.com](mailto:drott@google.com))

**Author:** Behdad Esfahbod ([behdad@behdad.org](mailto:behdad@behdad.org))

# Introduction

The Introduction section is expected be converted into specific OFF section edits later in the specification process

We propose extension of the COLR table to allow gradient fills in addition to the existing solid color fills. The current version number of COLR table is 0. We propose this as COLR table format version 1.

It is our understanding that this brings the capabilities of COLR/CPAL to match those of SVG Native for vector graphics. SVG Native allows embedding PNG and JPEG images while this proposal does not. We like to explore in the future, how COLR/CPAL can be mixed with sbix to address that limitation as well.

## High-level Design

COLR table is extended to expose a new vector of layers per glyph. If a glyph is not found in the new vector, client will try finding it in the COLR v0 glyph vector and fall back to no-color if the glyph is not found there either.

A glyph using the new extension is mapped to a list of layers. Each layer is formed by a directed acyclic graph of paints. Several different types of paint are defined:

1. Solid paints a solid color
2. Linear gradient paints a linear gradient
3. Radial gradient paints a radial gradient
4. Glyph draws a non-COLR glyph filled by some other paint
   * A COLR v1 glyph made up of a list of Glyph paints is equivalent to a COLR v0 Layer Record with the added ability to use gradients.
5. COLR Glyph reuses a COLR v1 glyph at a new position in the graph
6. Transformed reuses another paint, applying an affine transformation
7. Composite reuses two other paints, applying a compositing rule to combine them
   * We draw on <https://www.w3.org/TR/compositing-1/> for our composite modes

We have added an "alpha" (transparency) scalar to each invocation of a palette color. This allows for the expression of translucent versions of palette entries, as well as foreground, which we find useful. Without this, various translucent shades of the same color would need to be encoded separately in the color palette, which is undesirable since color palette entries are designed to be exposed to end-users.

All values expressed are *variable* by way of OFF Font Variations.

## Backwards Compatibility

The proposed design allows full backwards compatibility. This means, that a font designed for COLR format v1 specification, can contain sufficient information to be readable by a layout and rasterization engine that understands the v0 format. This is possible because the format version of the COLR table is a short format, as such considered a "minor", not a “major” version number.

If table format v1 is chosen, additional data will be read which specifies the additional information for gradients.

## Graphical Primitives

The two main graphical primitives that are added are linear gradients and radial gradients.

In most graphics systems, linear gradients are declared using two points, and radial gradients are declared using two circles. Such graphics systems also support a transformation matrix, via which one can get shear in linear gradients, or arbitrary ellipses with radial gradients. Since our proposed format does *not* have such universal transform underneath, the definition of linear and radial gradients are extended from their typical form to accommodate for these transformations in the gradient declaration itself.

### Color Line

A color line is a function that maps real numbers to a color value to define a 1-dimensional gradient, to be used and referenced from [Linear Gradients](https://github.com/googlefonts/colr-gradients-spec/tree/off_sub_1#linear-gradients) and [Radial Gradients](https://github.com/googlefonts/colr-gradients-spec/tree/off_sub_1#radial-gradients). Colors of the gradient are defined by *color stops*.

Color stops are defined at color stop positions. Color stop position 0 maps to the start point of a linear gradient or the center of the first circle of a radial gradient. Color stop position 1 maps to the end point of a linear gradient or the center of the second circle of a radial gradient. In the interval [0, 1] the color line must contain at least one color stop, but may contain multiple color stops that define the gradient.

Outside the defined interval, the gradient pattern in between the outer defined positions is repeated according to the color line [extend mode](https://github.com/googlefonts/colr-gradients-spec/tree/off_sub_1#heading=h.4dwrambuyuzf).

If there are multiple color stops defined for the same coordinate, the first one is used for computing the color value for values below the coordinate, the last one is used for computing the color value for values above. All other color stops for this coordinate are ignored.

Limiting the specified interval to a sub-range of [0, 1] allows for looping through colors repeatedly along the mapped distance, without having to encode them multiple times. In that sense, our color line is similar to CSS [repeating-linear-gradient()](https://developer.mozilla.org/en-US/docs/Web/CSS/repeating-linear-gradient) and [repeating-radial-gradient()](https://developer.mozilla.org/en-US/docs/Web/CSS/repeating-radial-gradient) functions.

In order to achieve:

* one gradient along the gradient positions (linear, or radial) and padded colors outside this range, color stops at 0 and 1 must be defined, and color line extend mode *pad* must be used. This achieves similarly behavior as defined in CSS [linear-gradient()](https://developer.mozilla.org/en-US/docs/Web/CSS/linear-gradient) and [radial-gradient()](https://developer.mozilla.org/en-US/docs/Web/CSS/radial-gradient) functions.
* a repeated gradient along the gradient positions (linear or radial): divide 1 by the number of desired repetitions and use the result as your maximum color stop, then use color line extend mode *repeat* to have it continue outside the defined interval.
* a mirrored / color-circle gradient: divide 1 by two times the number of desired full color stripes, and define the color stops between the 0 and the result of this division, then use color line extend mode *reflect* to have it continue mirrored.

*Figure 1: Repeating linear and radial gradients (*[*source*](https://cssnewbie.com/apply-cool-linear-and-radial-gradients-using-css/)*)*

#### Extend Mode

We propose three extend modes to control the behavior of the gradient outside its specified endpoints:

##### Extend Pad

For numbers outside the defined interval the color line continues to map to the outer color values, i.e. for values less than the leftmost defined color stop, it maps to the leftmost color stop value; for values greater than the rightmost defined color stop value, it maps to the rightmost defined color value.

##### Extend Repeat

For numbers outside the interval, the color line continues to map as if the defined interval was repeated.

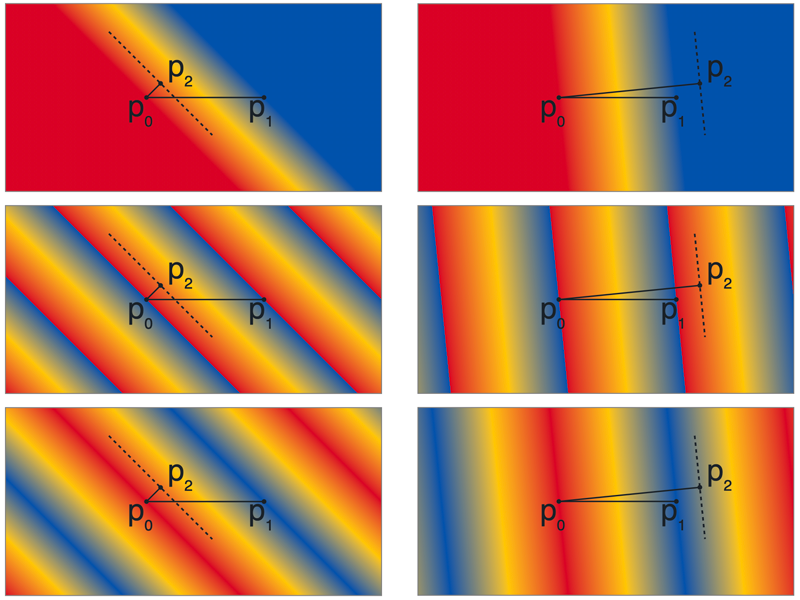
##### Extend Reflect

For numbers outside the defined interval, the color continues to map as if the interval would continue mirrored from the previous interval. This allows defining stripes in rotating colors.

### Linear Gradients

We propose definitions of linear gradients with two color line points P0 and P1 between which a gradient is interpolated. A point P2 is defined to rotate the gradient angle / orientation separately from the color line endpoints.

If the dot-product (P₁ - P₀) . (P₂ - P₀) is zero (or near-zero for an implementation-defined definition) then gradient is ill-formed and nothing must be rendered.



*Figure 2: Examples of linear gradients and their defining points with extend modes pad, repeat and reflect (top to bottom) with color stops for blue at 0, yellow at 0.5 and red at 1. (Illustration generated from* [*images/radial\_gradients.svg*](https://github.com/googlefonts/colr-gradients-spec/blob/off_sub_1/images/linear_gradients.html)*, requires glMatrix.js to work)*

### Radial Gradients

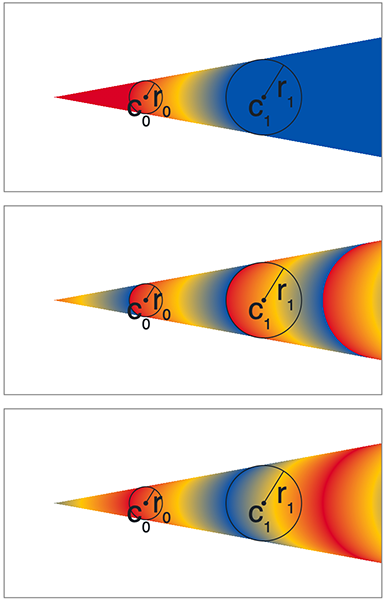
Radial gradients in this proposal are defined based on circles. If subject to a transform (via PaintTransformed) those circles may become ellipses.

A radial gradient in this proposal is a gradient between two—optionally transformed—circles, namely with center c0 and radius r0, and center c1 and radius r1 and a specified color line. The circle c0, r0 will be drawn with the color at color line position 0. The circle c1, r1 will be drawn with the color at color line colorLine position 1.

The drawing algorithm radial gradients follows the [HTML WHATWG Canvas spec for createRadialGradient()](https://html.spec.whatwg.org/multipage/canvas.html#dom-context-2d-createradialgradient). Quoting and adapting from there. With circle center points c0 and c1 defined as c0 = (x0, y0) and c1 = (x1, y1):

Radial gradients must be rendered by following these steps:

1. If c0 = c1 and r0 = r1 then the radial gradient must paint nothing. Return.
2. Let x(ω) = (x1-x0)ω + x0 Let y(ω) = (y1-y0)ω + y0 Let r(ω) = (r1-r0)ω + r0  
   Let the color at ω be the color at that position on the gradient color line (with the colors coming from the interpolation and extrapolation described above).
3. For all values of ω where r(ω) > 0, starting with the value of ω nearest to positive infinity and ending with the value of ω nearest to negative infinity, draw the circumference of the ellipse resulting from translating circle with radius r(ω) by affine transform at position (x(ω), y(ω)), with the color at ω, but only painting on the parts of the bitmap that have not yet been painted on by earlier circles in this step for this rendering of the gradient.



*Figure 3: Example of a radial gradient rendering with extend modes pad, repeat and reflect (top to bottom) with color stops for blue at 0, yellow at 0.5 and red at 1. (Illustration generated from* [*images/radial\_gradients.svg*](https://github.com/googlefonts/colr-gradients-spec/blob/off_sub_1/images/radial_gradients.svg)*)*

Note: Implementations must be careful to properly render radial gradient even if they are subject to a [*degenerate*](https://en.wikipedia.org/wiki/Invertible_matrix) or *near-degenerate* transform. Such radial gradients do have a well-defined shape, which is a strip or a cone filled with a linear gradient.

# OFF Changes

We're proposing changes to the following sections of ISO/IEC 14496-22:2019 “OFF”:

* 4.3 Data types
* 5.7.11 COLR – Color Table
* Bibliography

An overview of the design is provided, followed by the suggested specific changes.

## OFF 4.3 Data types

One new data type is proposed:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Offset24 | 24-bit offset to a table, same as uint24. NULL offset= 0x0000 |

## OFF 5.7.11 COLR – Color Table

The current header should be noted as *COLR version 0 header*.

A new section for *COLR version 1 header* should be added, along with a set of related records and tables.

### COLR v1 data structures

This section describes new and modified tables and records for COLR v1.

Offsets are always relative to the start of the containing struct.

#### Header, glyphs, layers

##### V1 Header

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint16 | version | Table version number—set to 1. |
| uint16 | numBaseGlyphRecords | May be 0 in a version 1 table. |
| Offset32 | baseGlyphRecordsOffset | Offset to baseGlyphRecords array (may be NULL). |
| Offset32 | layerRecordsOffset | Offset to layerRecords array (may be NULL). |
| uint16 | numLayerRecords | May be 0 in a version 1 table. |
| Offset32 | baseGlyphV1ListOffset | Offset to BaseGlyphV1List table. |
| Offset32 | itemVariationStoreOffset | Offset to ItemVariationStore (may be NULL). |

##### BaseGlyphV1List table

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint32 | numBaseGlyphV1Records |  |
| BaseGlyphV1Record | baseGlyphV1Records[numBaseGlyphV1Records] |  |

##### BaseGlyphV1Record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | glyphID | Glyph ID of the base glyph. |
| Offset32 | layerListOffset | Offset to LayerV1List table, from start of BaseGlyphsV1List table. |

*Note:* The glyph ID is not limited to the numGlyphs value in the 'maxp' table.

##### LayerV1List table

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | numLayers |  |
| Offset32 | paintOffset[numLayers] | Offsets to Paint tables, from the start of the LayerV1List table. |

*Note:* For large layer counts PaintComposite can be used to combine multiple COLR v1 glyphs.

#### Variation structures

The following records are defined to facilitate COLR v1 font variation support.

To indicate no variation, set varOuterIndex and varInnerIndex to 0xFFFF.

##### VarFWord record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| FWORD | coordinate |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

##### VarUFWord record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| UFWORD | distance |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

##### VarFixed record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Fixed | value |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

*Note:* In order to combine deltas with Fixed values, the ItemVariationStore format is extended to allow for int32 deltas. When combining a Fixed value with 32-bit deltas, the Fixed value is treated as though it were int32.

##### VarF2Dot14 record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| F2Dot14 | value |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

Values are inherently limited to [-2., 2). In some contexts, limited to the [-1., 1.] or [0., 1.].

*Note:* When combining an F2Dot14 with 16-bit deltas, the F2Dot14 is treated as though it were int16.

#### Color structures

##### Extend enumeration

|  |  |  |
| --- | --- | --- |
| **Value** | **Name** | **Description** |
| 0 | EXTEND\_PAD | Use nearest color stop. |
| 1 | EXTEND\_REPEAT | Repeat from farthest color stop. |
| 2 | EXTEND\_REFLECT | Mirror color line from nearest end. |

If a ColorLine.extend value is not recognized, use EXTEND\_PAD.

##### ColorIndex record

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

Values for alpha outside [0.,1.] are reserved.

The ColorIndex alpha is multiplied into the alpha of the CPAL entry (converted to float -- divide by 255) to produce a final alpha.

##### ColorStop record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarF2Dot14 | stopOffset | Proportional distance on a color line; variable. |
| ColorIndex | color |  |

##### ColorLine table

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

#### Paint structures

##### PaintSolid table (format 1)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 1. |
| ColorIndex | color | Solid color fill. |

##### PaintLinearGradient table (format 2)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 2. |
| Offset24 | colorLineOffset | Offset to ColorLine, from start of PaintLinearGradient table. |
| VarFWord | x0 | Start point x coordinate. |
| VarFWord | y0 | Start point y coordinate. |
| VarFWord | x1 | End point x coordinate. |
| VarFWord | y1 | End point y coordinate. |
| VarFWord | x2 | Rotation vector end point x coordinate. |
| VarFWord | y2 | Rotation vector end point y coordinate. |

For linear gradient without skew, set x2,y2 to x1,y1.

##### PaintRadialGradient table (format 3)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint16 | format | set to 3 |
| Offset32 | colorLineOffset | offset from start of PaintRadialGradient 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 |

##### PaintGlyph table (format 4)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 4. |
| Offset24 | paintOffset | Offset to a Paint table, from start of PaintGlyph table. |
| uint16 | glyphID | Glyph ID for the source outline. |

Glyph outline is used as clip mask for the content in the Paint subtable. Glyph ID must be less than the numGlyphs value in the 'maxp' table.

##### PaintColrGlyph table (format 5)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 5. |
| uint16 | glyphID | Virtual glyph ID for a BaseGlyphV1List base glyph. |

Glyph ID must be in the BaseGlyphV1List; may be greater than maxp.numGlyphs.

##### PaintTransformed table (format 6)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 6. |
| Offset24 | paintOffset | Offset to a Paint subtable, from start of PaintTransformed table. |
| Affine2x3 | transform | An Affine2x3 record (inline). |

##### PaintComposite table (format 7)

|  |  |  |
| --- | --- | --- |
| **Type** | **Field name** | **Description** |
| uint8 | format | Set to 7. |
| Offset24 | sourcePaintOffset | Offset to a source Paint table, from start of PaintComposite table. |
| uint8 | compositeMode | A CompositeMode enumeration value. |
| Offset24 | backdropPaintOffset | Offset to a backdrop Paint table, from start of PaintComposite table. |

If compositeMode value is not recognized, COMPOSITE\_CLEAR is used.

#### Composite modes

Supported composition modes are taken from the W3C [Compositing and Blending Level 1](https://www.w3.org/TR/compositing-1/) specification.

##### CompositeMode enumeration

|  |  |  |
| --- | --- | --- |
| **Value** | **Name** | **Description** |
|  | *Porter-Duff modes* |  |
| 0 | COMPOSITE\_CLEAR | See [Clear](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_clear) |
| 1 | COMPOSITE\_SRC | See [Copy](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_src) |
| 2 | COMPOSITE\_DEST | See [Destination](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dst) |
| 3 | COMPOSITE\_SRC\_OVER | See [Source Over](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcover) |
| 4 | COMPOSITE\_DEST\_OVER | See [Destination Over](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstover) |
| 5 | COMPOSITE\_SRC\_IN | See [Source In](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcin) |
| 6 | COMPOSITE\_DEST\_IN | See [Destination In](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstin) |
| 7 | COMPOSITE\_SRC\_OUT | See [Source Out](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcout) |
| 8 | COMPOSITE\_DEST\_OUT | See [Destination Out](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstout) |
| 9 | COMPOSITE\_SRC\_ATOP | See [Source Atop](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcatop) |
| 10 | COMPOSITE\_DEST\_ATOP | See [Destination Atop](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstatop) |
| 11 | COMPOSITE\_XOR | See [XOR](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_xor) |
|  | *Separable color blend modes:* |  |
| 12 | COMPOSITE\_SCREEN | See [screen blend mode](https://www.w3.org/TR/compositing-1/#blendingscreen) |
| 13 | COMPOSITE\_OVERLAY | See [overlay blend mode](https://www.w3.org/TR/compositing-1/#blendingoverlay) |
| 14 | COMPOSITE\_DARKEN | See [darken blend mode](https://www.w3.org/TR/compositing-1/#blendingdarken) |
| 15 | COMPOSITE\_LIGHTEN | See [lighten blend mode](https://www.w3.org/TR/compositing-1/#blendinglighten) |
| 16 | COMPOSITE\_COLOR\_DODGE | See [color-dodge blend mode](https://www.w3.org/TR/compositing-1/#blendingcolordodge) |
| 17 | COMPOSITE\_COLOR\_BURN | See [color-burn blend mode](https://www.w3.org/TR/compositing-1/#blendingcolorburn) |
| 18 | COMPOSITE\_HARD\_LIGHT | See [hard-light blend mode](https://www.w3.org/TR/compositing-1/#blendinghardlight) |
| 19 | COMPOSITE\_SOFT\_LIGHT | See [soft-light blend mode](https://www.w3.org/TR/compositing-1/#blendingsoftlight) |
| 20 | COMPOSITE\_DIFFERENCE | See [difference blend mode](https://www.w3.org/TR/compositing-1/#blendingdifference) |
| 21 | COMPOSITE\_EXCLUSION | See [exclusion blend mode](https://www.w3.org/TR/compositing-1/#blendingexclusion) |
| 22 | COMPOSITE\_MULTIPLY | See [multiply blend mode](https://www.w3.org/TR/compositing-1/#blendingmultiply) |
|  | *Non-separable color blend modes:* |  |
| 23 | COMPOSITE\_HSL\_HUE | See [hue blend mode](https://www.w3.org/TR/compositing-1/#blendinghue) |
| 24 | COMPOSITE\_HSL\_SATURATION | See [saturation blend mode](https://www.w3.org/TR/compositing-1/#blendingsaturation) |
| 25 | COMPOSITE\_HSL\_COLOR | See [color blend mode](https://www.w3.org/TR/compositing-1/#blendingcolor) |
| 26 | COMPOSITE\_HSL\_LUMINOSITY | See [luminosity blend mode](https://www.w3.org/TR/compositing-1/#blendingluminosity) |

#### Transform

##### Affine2x3 record

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarFixed | xx |  |
| VarFixed | xy |  |
| VarFixed | yx |  |
| VarFixed | yy |  |
| VarFixed | dx | Translation in x direction. |
| VarFixed | dy | Translation in y direction. |

This is a standard 2x3 matrix for 2D affine transformation.

#### Constraints

Constraints on the data structures making up a COLR version 1 should be noted.

##### Acyclic Graphs Only

PaintColrGlyph allows recursive composition of COLR glyphs. This is desirable for reusable parts but introduces the possibility of a cyclic graph. Implementations should track the COLR gids they have seen in processing and fail if a gid is reached repeatedly.

##### Bounded Layers Only

Every entry in the LayerV1List must define a bounded region. Implementations must confirm this invariant. Unbounded layers must not render.

The following paints are always bounded:

* PaintGlyph
* PaintColrGlyph

The following paints are always unbounded:

* PaintSolid
* PaintLinearGradient
* PaintRadialGradient

The following paints *may* be bounded:

* PaintTransformed is bounded IFF the source is bounded
* PaintComposite boundedness varies by mode:
  + Always bounded
    - COMPOSITE\_CLEAR
  + Bounded IFF src is bounded
    - COMPOSITE\_SRC
    - COMPOSITE\_SRC\_OUT
  + Bounded IFF backdrop is bounded
    - COMPOSITE\_DEST
    - COMPOSITE\_DEST\_OUT
  + Bounded IFF src OR backdrop is bounded
    - COMPOSITE\_SRC\_IN
    - COMPOSITE\_DEST\_IN
  + Bounded IFF src AND backdrop are bounded
    - *all other modes*

##### Bounding Box

The bounding box of the base (non-COLR) glyph referenced from the BaseGlyphV1Record (by BaseGlyphV1Record::gid) should be taken to describe the bounding box for the COLR v1 glyph.

Note: A glyf entry with two points at the diagonal extrema would suffice.

Note: This can be used to allocate a drawing surface without traversing the COLR v1 glyph structure.

#### Understanding COLR v1

Addition of explanatory content explaining how COLR version 1 functions should be added.

##### Alpha

The alpha channel for a layer can be populated using PaintComposite:

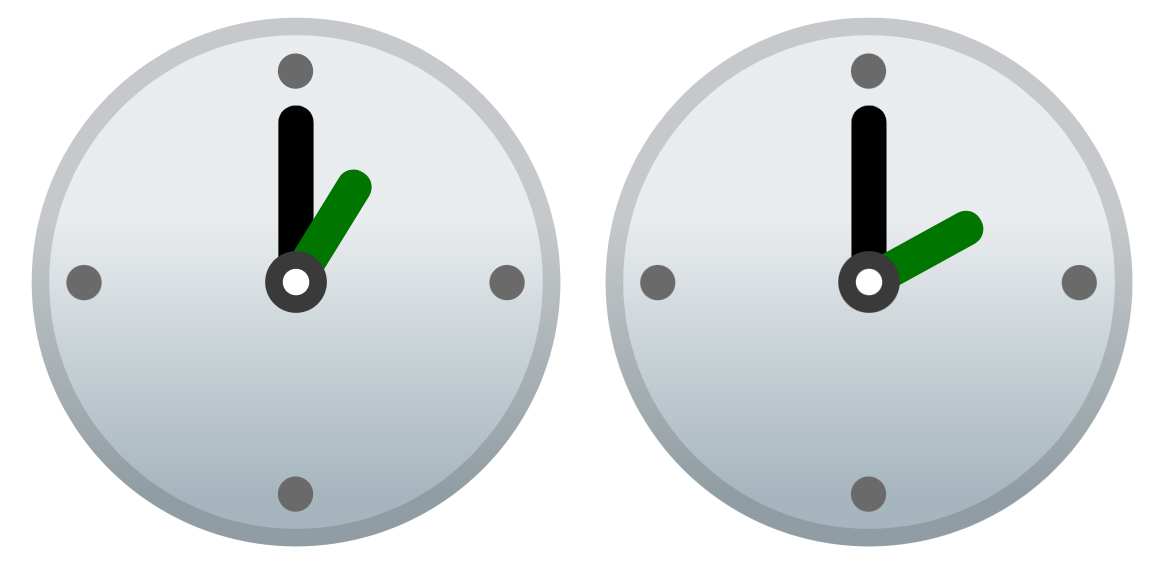
* PaintSolid can be used to set a blanket alpha
* PaindLinearGradient and PaintRadialGradient can be used to set gradient alpha
* Mode [Source In](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcin) can be used to mask

##### Reusable Parts

Use PaintTransformed to reuse parts in different positions or sizes.

Use PaintColrGlyph to reuse entire COLR glyphs.

For example, consider the Noto clock emoji (hand colored for emphasis):



The entire backdrop (outline, gradient-circle, 4 dots, the minute hand) is reusable for all versions of the clock:



The hour hand is reusable as a transformed glyph.

Another example might be emoji faces: many have the same backdrop with different eyes, noses, tears, etc drawn on top.

## Bibliography

Add references to:

* <https://www.w3.org/TR/compositing-1/>.