OpenVG™ API 1.1 Quick Reference Card

OpenVG (Open Vecter Graphics) is an application programming interface (API) for hardware accelerated two-dimensional vector and raster graphics developed under the auspices of the Khronos Group . It provides a device independent and vendor-neutral interface for sophisticated 2D graphical applications, while allowing device manufacturers to provide hardware acceleration where appropriate.

[n.n.n] refer to the section in the API Sepcification available at www.khronos.org/openvg.

Data Type

Primitive Data Type [3.2]

openvg.h	khronos_type.h	
VGbyte	khronos_int8_t	*[-128, 127]*
VGubyte	khronos_uint8_t	*[0, 256]*
VGshort	khronos_int16_t	*[-32768, 32767]*
VGushort	khronos_int16_t	*[0, 65535]*
VGint	khronos_int32_t	*[-2^{31}, 2^{31}-1]*
VGuint	khronos_int32_t	*[0, (2^{32}-1)]*
vg_bitfield	khronos_uint32_t	*Same as VGuint*
vg_boolean	khronos_int32_t	*0,1*
vg_float	khronos_float_t	*IEEE 754 Standard*

Number Representations [3.3]

VG_MAXSHORT largest positive value of VGshort, smallest negative value is (–VG_MAXSHORT – 1)
VG_MAXINT largest positive value of VGint, smallest negative value is (–VG_MAXINT – 1)
VG_MAX_FLOAT largest floating-point number

Handle Based Type [3.6]

typedef VGuint VGHandle

VGFont	reference to font data
VGImage	reference to image data
VGMaskLayer	reference to mask data
VGPaint	reference to a paint specification
VGPath	reference to path data

EGL 1.3 Functions [4.2]

EGLBoolean **eglBindAPI**(EGLenum *api*)
api should be EGL_OPENVG_API to bind OpenVG use
EGLContext **eglCreateContext** (EGLDisplay *dpy*, EGLConfig *config*, EGLContext *share_context*, const EGLint* *attrib_list*)
EGLSurface **eglCreateWindowSurface** (EGLDisplay *dpy*, EGLConfig *config*, NativeWindowType *win*, const EGLint* *attrib_list*);
attrib_list is array with pairs of param_name and value. last entity of array must be EGL_NONE
EGLSurface **eglCreatePbufferFromClientBuffer** (EGLDisplay *dpy*, EGLenum buftype, EGLClientBuffer buffer, EGLConfig config, const EGLint* *attrib_list*)
Pbuffer(off-screen buffer) allow rendering into a VGImage
EGLBoolean **eglMakeCurrent** (EGLDisplay *dpy*, EGLSurface *draw*, EGLSurface *read*, EGLContext *context*)
become current on the running thread
EGLContext **eglGetCurrentContext** ()
get Current Context
EGLBoolean **eglDestroyContext** (EGLDisplay *dpy*, EGLContext *context*)
EGLBoolean **eglSwapBuffers** (EGLDisplay *dpy*, EGLSurface *surface*);

param_name: Config Attributes

EGL_BUFFER_SIZE	EGL_ALPHA_SIZE
EGL_BLUE_SIZE	EGL_GREEN_SIZE
EGL_RED_SIZE	EGL_DEPTH_SIZE
EGL_STENCIL_SIZE	EGL_CONFIG_CAVEAT
EGL_CONFIG_ID	EGL_LEVEL
EGL_MAX_PBUFFER_HEIGHT	
EGL_MAX_PBUFFER_PIXELS	
EGL_MAX_PBUFFER_WIDTH	
EGL_NATIVE_RENDERABLE	
EGL_NATIVE_VISUAL_ID	
EGL_NATIVE_VISUAL_TYPE	
EGL_PRESERVED_RESOURCES	
EGL_SAMPLES	EGL_SAMPLE_BUFFERS
EGL_SURFACE_TYPE	EGL_TRANSPARENT_TYPE
EGL_TRANSPARENT_BLUE_VALUE	
EGL_TRANSPARENT_GREEN_VALUE	
EGL_TRANSPARENT_RED_VALUE	
EGL_NONE	EGL_BIND_TO_TEXTURE_RGB
EGL_BIND_TO_TEXTURE_RGBA	
EGL_MIN_SWAP_INTERVAL	
EGL_MAX_SWAP_INTERVAL	
EGL_LUMINANCE_SIZE	EGL_ALPHA_MASK_SIZE
EGL_COLOR_BUFFER_TYPE	
EGL_RENDERABLE_TYPE	
EGL_MATCH_NATIVE_PIXMAP	
EGL_CONFORMANT	EGL_CONFORMANT_KHR
EGL_CONFORMANT	EGL_SLOW_CONFIG
EGL_NON_CONFORMANT_CONFIG	
EGL_TRANSPARENT_RGB	
EGL_RGB_BUFFER	
EGL_LUMINANCE_BUFFER	
EGL_NO_TEXTURE	EGL_TEXTURE_RGB
EGL_TEXTURE_RGBA	EGL_TEXTURE_2D
EGL_PBUFFER_BIT	EGL_PIXMAP_BIT
EGL_WINDOW_BIT	
EGL_VG_COLORSPACE_LINEAR_BIT	
EGL_VG_ALPHA_FORMAT_PRE_BIT	
EGL_OPENGL_ES_BIT	EGL_OPENVG_BIT

API for Forcing Drawing to Complete [4.3]

void **vgFlush** (void) complete request in finite time
void **vgFinish** (void) complete request

To Use OpenVG, EGL_OPENVG_BIT should be set in EGL_RENDERABLE_TYPE (read-only parameter)
EGL_ALPHA_MASK_SIZE bit depth of the mask. If value is 0, masking is disabled.

Error [4.1]

VGErrorCode **vgGetError** (void) : *return oldest error code*

VGErrorCode

VG_NO_ERROR	0
VG_BAD_HANDLE_ERROR	0x1000
VG_ILLEGAL_ARGUMENT_ERROR	0x1001
VG_OUT_OF_MEMORY_ERROR	0x1002
VG_PATH_CAPABILITY_ERROR	0x1003
VG_UNSUPPORTED_IMAGE_FORMAT_ERROR	0x1004
VG_UNSUPPORTED_PATH_FORMAT_ERROR	0x1005
VG_IMAGE_IN_USE_ERROR	0x1006
VG_NO_CONTEXT_ERROR	0x1007

VGU

V is void, F is VGfloat, S is VGshort, N is VGint, P is VGImage or VGPath, B is VGboolean, E is vguErrorCode

High-Level Geomteric Primitives [17.1]

E **vguLine** (P *path*, F *x0*, F *y0*, F *x1*, F *y1*)
E **vguPolygon** (P *path*, const F* *points*, N *count*, B *closed*)
E **vguRect** (P *path*, F *x*, F *y*, F *w*, F *h*)
E **vguRoundRect** (P *path*, F *x*, F *y*, F *w*, F *h*, F *arcW*, F *arcH*)
E **vguEllips** (P *path*, F *cx*, F *cy*, F *w*, F *h*)
E **vguArc** (P *path*, F *x*, F *y*, F *w*, F *h*, F *startAngle*, F *angleExt*, VGUArcType *arcType*)
VGUArcType
VGU_ARC_OPEN
VGU_ARC_CHORD
VGU_ARC_PIE

Image Warping [17.2]

E **vguComputeWarpQuadToSquare** (F *sx0*, F *sy0*, F *sx1*, F *sy1*, F *sx2*, F *sy2*, F *sx3*, F *sy3*, F* *matrix*)
E **vguComputeWarpSquareToQuad** (F *dx0*, F *dy0*, F *dx1*, F *dy1*, F *dx2*, F *dy2*, F *dx3*, F *dy3*, F* *matrix*)
E **vguComputeWarpQuadToQuad** (F *sx0*, F *sy0*, F *sx1*, F *sy1*, F *sx2*, F *sy2*, F *sx3*, F *sy3*, F *dx0*, F *dy0*, F *dx1*, F *dy1*, F *dx2*, F *dy2*, F *dx3*, F *dy3*, F* *matrix*)

Object Parameter Set/Get [5.3]

V is void, F is VGfloat, N is VGint, H is VGHandle
V **vgSetParameterf** (H *obj*, N *type*, F *val*)
V **vgSetParameteri** (H *obj*, N *type*, F *val*)
V **vgSetParameterfv** (H *obj*, N *type*, N *cnt*, const F * *val*)
V **vgSetParameteriv** (H *obj*, N *type*, N *cnt*, const N * *val*)

F **vgGetParameterf** (H *obj*, N *type*)
N **vgGetParameteri** (H *obj*, N *type*)
N **vgGetParameterVectorSize**(H *obj*, N *type*)
V **vgGetParameterfv**(H *obj*, N *type*, N *cnt*, F * *val*)
V **vgGetParameteriv**(H *obj*, N *type*, N *cnt*, N * *val*)

Font

Parameter of Paint Object [10.4]

VGFontParamType
blue text *is default value, ( ) is data type of parameter*
VG_FONT_NUM_GLYPHS (VGInt)
0

Draw Text [11.5]

V **vgDrawGlyph** (P *font*, N *glyphIndex*, B *paintModes*, vgboolean allowAutoHinting)
V **vgDrawGlyphs** (P *font*, N *glyphCount*, const N* *glyphIndices*, const F* *adjustment_x*, const_F* *adjustment_y*, B *paintModes*, vgboolean *allowAutoHinting*)
// paintModes are VG_FILL_PATH and or VG_STOKE_PATH

Fonts Operations

V is void, F is VGfloat, N is VGint, P is VGFont, B is VGbitfield

Font Definition [11.4]

typedef VGHandle VGFont;

Managing VGFont Object [11.4.2]

P **vgCreateFont** (N *glyphCapacityHint*)
V **vgDestroyFont** (P *font*)

Adding and Modifying Glyphs in Fonts [11.4.4]

V **vgSetGlyphToPath** (P *font*, N *glyphIndex*, VGPath *path*, vgBoolean inHinted, const F *origin[2]*, const F *escape[2]*)
V **vgSetGlyphToImage** (P *font*, N *glyphIndex*, VGImage *image*, const F *origin[2]*, const F *escape[2]*)
V **vgClearGlyph** (P *font*, N *glyphIndex*)

KHRONOS GROUP

The Khronos Group is an industry consortium creating open standards for the authoring and acceleration of parallel computing, graphics and dynamic media on a wide variety of platforms and devices. See www.khronos.org to learn more about the Khronos Group.

OpenVG is a trademark of Khronos Group.

See www.khronos.org/openvg for the full specification.

Path

Segment Commands [8.5.2]

VG_RELATIVE command is VG_ABSOLUTE command + 1
In VG_RELATIVE , x0,y0 ~ x2,y2 is relative position from (ox,oy)

VGPathSegment	Coordinates	Val	Description (Side Effects)
VG_CLOSE_PATH	none	0	*(px,py)=(ox,oy)=(sx,sy)* *End current subpath*
VG_MOVE_TO	x0,y0	2	*(sx,sy)=(px,py)=(ox,oy)=(x0,y0)* *End current subpath*
VG_LINE_TO	x0,y0	4	*(px,py)=(ox,oy)=(x0,y0)*
VG_HLINE_TO	x0	6	*y0=oy, (px,py)=(x0,oy), ox=x0*
VG_VLINE_TO	y0	8	*x0=ox, (px,py)=(ox,y0), oy=y0*
VG_QUAD_TO	x0,y0,x1,y1	10	*(px,py)=(x0,y0), (ox,oy)=(x1,y1)*
VG_CUBIC_TO	x0,y0 ~ x2,y2	12	*(px,py)=(x1,y1), (ox,oy)=(x2,y2)*
VG_SQUAD_TO	x1,y1	14	*(x0,y0)=(2*ox-px,2*oy-py)* *(px,py)=(x0,y0),(ox,oy)=(x1,y1)*
VG_SCUBIC_TO	x1,y1,x2,y2	16	*(x0,y0)=(2*ox-px,2*oy-py)* *(px,py)=(x1,y1),(ox,oy)=(x2,y2)*
VG_SCCWARC_TO	rh,rv,rot,x0,y0	18	*(px,py)=(ox,oy)=(x0,y0)*
VG_SCWARC_TO	rh,rv,rot,x0,y0	20	*(px,py)=(ox,oy)=(x0,y0)*
VG_LCCWARC_TO	rh,rv,rot,x0,y0	22	*(px,py)=(ox,oy)=(x0,y0)*
VG_LCWARC_TO	rh,rv,rot,x0,y0	24	*(px,py)=(ox,oy)=(x0,y0)*

Path Data Example [8.5.5]

Triangle Shape

```
VGubyte segments[] = { VG_MOVE_TO_ABS,VG_LINE_TO,VG_LINE_TO,
        VG_CLOSE_PATH };
VGfloat coords[] = { 50.0f, 100.0f, 190.0f, 100.0f, 120.0f, 240.0f };
VGPath path = vgCreatePath(VG_PATH_FORMAT_STANDARD,
    VG_PATH_DATATYPE_F,  1.0f, 0.0f, 0, 0, VG_PATH_CAPABILITY_ALL);
vgAppendPathData( path, 4, segments, coords );
```

Parameter of Path Object [8.5.3]

blue text *is default value,*
( ) is data type of parameter

VG_PATH_FORMAT (VGint)
- VG_PATH_FORMAT_STANDARD 0

VG_PATH_DATATYPE (VGPathDatatype)
- VG_PATH_DATATYPE_S_8
- VG_PATH_DATATYPE_S_16
- VG_PATH_DATATYPE_S_32
- VG_PATH_DATATYPE_F

VG_PATH_SCALE (VGfloat) 1.0f
VG_PATH_BIAS (VGfloat) 0.0f
VG_PATH_NUM_SEGMENTS (VGint) 0
VG_PATH_NUM_COORDS (VGint) 0

VGPathCapabilities
- VG_PATH_CAPABILITY_APPEND_FROM
- VG_PATH_CAPABILITY_APPEND_TO
- VG_PATH_CAPABILITY_MODIFY
- VG_PATH_CAPABILITY_TRANSFORM_FROM
- VG_PATH_CAPABILITY_TRANSFORM_TO
- VG_PATH_CAPABILITY_INTERPOLATE_FROM
- VG_PATH_CAPABILITY_INTERPOLATE_TO
- VG_PATH_CAPABILITY_PATH_LENGTH
- VG_PATH_CAPABILITY_POINT_ALONG_PATH
- VG_PATH_CAPABILITY_TANGENT_ALONG_PATH
- VG_PATH_CAPABILITY_PATH_BOUNDS
- VG_PATH_CAPABILITY_PATH_TRANSFORMED_BOUNDS
- VG_PATH_CAPABILITY_ALL

Draw Path [8.8]

V **vgDrawPath** (P *path*, B *paintModes*)
// paintModes are VG_FILL_PATH and or
// VG_STOKE_PATH

Path Operations [8.6]

V is void, F is VGfloat, N is VGint, P is VGPath, B is VGbitfield

Create and Destroy Path [8.6.2]

P **vgCreatePath**(N *pathFormat*, VGPathDatatype *datatype*, F *scale*, F *bias*, N *segCapacityHint*, N *coordCapacityHint*, B *capabilities*)
V **vgClearPath** (P *path*, VGbitfield *capabilities*)
V **vgDestroyPath** (P *path*)

Query and Modify Path Capabilities [8.6.4]

B **vgGetPathCapabilities**(P *path*)
V **vgRemovePathCapabilities** (P *path*, B *capabilities*)

Copying Data Between Paths [8.6.5]

V **vgAppendPath** (P *dstPath*, P *srcPath*)
V **vgAppendPathData** (P *dstPath*, N *numSeg*, const VGubyte *pathSeg, const V *pathData)

Modifying Path Data [8.6.7]

V **vgModifyPathCoords** (P *dstPath*, N *startIdx*, N *numSeg*, const V *pathData)

Transform Path [8.6.8]

(VG_MATRIX_PATH_USER_TO_SURFACE is applied)
V **vgTransformPath** (P *dstPath*, P *srcPath*)

Interpolating Between Paths [8.6.9]

VGboolean **vgInterpolatePath**(P *dstPath*, P *startPath*, P *endPath*, F *amount*)

Length of Path [8.6.10]

F **vgPathLength** (P *path*, N *startSeg*, N *numSeg*)

Position and Tangent Along Path [8.6.11]

V **vgPointAlongPath** (P *path*, N *startSeg*, N *numSeg*, F distance, F* *x*, F* *y*, F* *tanX*, F* *tanY*)

Querying Bounding Box [8.6.12]

V **vgPathBounds** (P *path*, F* *x*, F* *y*, F* *w*, F* *h*)
V **vgPathTransformedBounds** (P *path*, F* *x*, F* *y*, F* *w*, F* *h*)

Image

Image Definition [10.3]

typedef VGHandle VGImage;
VGImageQuality
- VG_IMAGE_QUALITY_NONANTIALIASED
- VG_IMAGE_QUALITY_FASTER
- VG_IMAGE_QUALITY_BETTER

Parameter of Image Object [10.4]

VGImageParamType
blue text *is default value, ( ) is data type of param*

VG_IMAGE_FORMAT (VGImageFormat)
- VG_s{RGBX,XRGB,BGRX,XBGR}_8888
- VG_s{RGBX,XRGB,BGRX,XBGR}_8888
- VG_s{RGBX,XRGB,BGRX,XBGR}_8888_PRE
- VG_s{RGBX,XRGB,BGRX,XBGR}_{1555,5551}
- VG_{RGBX,XRGB,BGRX,XBGR}_4444
- VG_{RGBX,XRGB,BGRX,XBGR}_8888
- VG_{RGBX,XRGB,BGRX,XBGR}_8888
- VG_{RGBX,XRGB,BGRX,XBGR}_8888_PRE
- VG_sL_8
- VG_lL_8
- VG_A_8
- VG_BW_1
- VG_A_1
- VG_A_4

VG_IMAGE_WIDTH (VGint) 0
VG_IMAGE_HEIGHT (VGint) 0

Image and Surface Operations

V is void, F is VGfloat, N is VGint, P is VGImage, B is VGbitfield

Create and Destroy Image [10.3]

P **vgCreateImage**(VGImageFormat *fmt*, N *w*, N *h*, B *quality*)
V **vgDestroyImage** (P *image*)

Reading and Writing Image Pixels [10.5]

V **vgClearImage** (P *image*, N *x*, N *y*, N *w*, N *h*)
V **vgImageSubData** (P *image*, const V* *data*, N *dataStride*, T *fmt*, N *x*, N *y*, N *w*, N *h*)
V **vgGetImageSubData** (P *image*, V* *data*, N *dataStride*, T *fmt*, N *x*, N *y*, N *w*, N *h*)

Child Images [10.6]

P **vgChildImage** (P *parent*, N *x*, N *y*, N *w*, N *h*)
P **vgGetParent** (P *image*)

Copying between Images [10.7]

V **vgCopyImage** (P *dst*, N *dx*, N *dy*, P *src*, N *sx*, N *sy*, N *w*, N *h*, VGboolean *dither*)

Draw Image [10.8]

V **vgDrawImage**(P *image*)

Reading and Writing Drawing Surface [10.9]

V **vgSetPixel** (N *dx*, N *dy*, P *src*, N *sx*, N *sy*, N *w*, N *h*)
V **vgWritePixels** (const V* *data*, N *dataStride*, T *fmt*, N *dx*, N *dy*, N *w*, N *h*)
V **vgGetPixels** (P *dst*, N *dx*, N *dy*, N *sx*, N *sy*, N *w*, N *h*)
V **vgReadPixels** (V* *data*, N *dataStride*, T *fmt*, N sx, N sy, N *w*, N *h*)
V **vgCopyPixel** (N *dx*, N *dy*, N *sx*, N *sy*, N *w*, N *h*)

Image Filter [12]

V is void, F is VGfloat, S is VGshort, N is VGint,
P is VGImage, B is VGbitfield, T is VGImageFormat

Color Combination [12.3]

(4x4 Color Matrix Multiplication)
V **vgColorMatrix** (P *dst*, P *src*, const F* *matrix*)

Convolution [12.4]

V **vgConvolve** (P *dst*, P *src*, N *kernelW*, N *KernelH*, N *shiftX*, N *shifty*, const S* *kernel*, F *scale*, F *bias*, VGTilingMode *tilingMode*)
V **vgSeparableConvolve** (P *dst*, P *src*, N *kernelW*, N *KernelH*, N *shiftX*, N *shifty*, const S* *kernelX*, cont S* *kernelY*, F *scale*, F *bias*, VGTilingMode *tilingMode*)
V **vgGaussianBlur** (P *dst*, P *src*, F *stdDevX*, F *stdDevY*, VGTilingMode *tilingMode*)

Lookup Table [12.5]

V **vgLookup** (P *dst*, P *src*, const VGubyte* *redLUT*, const VGubyte* *greenLUT*, const VGubyte* *blueLUT*, const VGubyte* *alphaLUT*, VGboolean *outputLinear* VGboolean *outputPremultiplied*)
V **vgLookupSingle** (P *dst*, P *src*, const VGuint* *LUT*, VGboolean *outputLinear*, VGboolean *outputPremultiplied*)

Src\Dst	Memory	Image	Surface
Memory	-	*vgImageSubData*	*vgWritePixels*
Image	*vgGetImageSubData*	*vgCopyImage*	*vgSetPixels*
Surface	*vgReadPixels*	*vgGetPixels*	*vgCopyPixels*

Paint

Parameter of Paint Object [10.4]

VGPaintParamType
blue text *is default value, ( ) is data type of parameter*

VG_PAINT_TYPE (VGPaintType)
- VG_PAINT_TYPE_COLOR
- VG_PAINT_TYPE_LINEAR_GRADIENT
- VG_PAINT_TYPE_RADIAL_GRADIENT
- VG_PAINT_TYPE_PATTERN

VG_PAINT_COLOR (VGfloat[4])
- {0.0f, 0.0f, 0.0f, 0.0f} // {red,green,blue,alpha}

VG_PAINT_COLOR_RAMP_SPREAD_MODE (VGColorRampSpreadMode)
- VG_COLOR_RAMP_SPREAD_PAD
- VG_COLOR_RAMP_SPREAD_REPEAT
- VG_COLOR_RAMP_SPREAD_REFRECT

VG_PAINT_COLOR_RAMP_PREMULTIPLIED (VGboolean)
- VG_TRUE
- VG_FALSE (disabled)

VG_PAINT_COLOR_RAMP_STOPS (VGfloat *)
- Null //{stop0, red0, green0, blue0, alpha0,.....}

VG_PAINT_LINEAR_GRADIENT (VGFloat[4])
- {0.0f, 0.0f, 1.0f, 0.0f}
- {startx, starty, endx, endy}

VG_PAINT_RADIAL_GRADIENT (VGFloat[5])
- {0.0f, 0.0f, 0.0f, 0.0f, 1.0f}
- {centerx, centery, focusx, focusy, radius}

VG_PAINT_PATTERN_TILING_MODE (VGTilingMode)
- VG_TILE_FILL
- VG_TILE_PAD
- VG_TILE_REPEAT
- VG_TILE_REFLECT

Paint Operations

V is void, F is VGfloat, N is VGint, P is VGPaint, B is VGbitfield

Paint Definition [9.1]

typedef VGHandle VGPaint;

Create and Destroy Paint [9.1.1]

P **vgCreatePaint**(void)
V **vgDestroyPaint** (P *paint*)

Setting the Current Paint [9.1.2]

V **vgSetPaint**(P *paint*, B *paintModes*)
P **vgGetPaint**(VGPaintMode *paintModes*)
// paintModes are VG_FILL_PATH and or VG_STOKE_PATH

Color Paint [9.2]

V **vgSetParameterfv** (P *paint*, VG_PAINT_COLOR, 4, F *col[4]*)
V **vgSetColor** (P *paint*, VGuint *rgba*)
VGuint **vgGetColor** (P *paint*)

OpenVG ™ API 1.1 Quick Reference Card: Drawing Context

Drawing Context [4]

Drawing Surface	Affine transformations for paint applied to geometry
Matrix Mode	Surface for drawing
Path user-to-surface Transformation	Affine transformation for filled and stroked geometry
Image user-to-surface Transform	Affine or projective transformation for images
Paint-to-user Transformations	Affine transformations for paint applied to geometry
Glyph user-to-surface Transformation	Affine transformation for glyphs
Glyph origin	(X,Y) origin of a glyph to be drawn
Fill Rule	Rule for filling paths
Quality Settings	Image and rendering quality, pixel layout
Color Transformation	Color Transformation Function
Blend Mode	Pixel blend function
Image Mode	Image/paint combination function
Scissoring	Current scissoring rectangles and enable/disable
Stroke	Stroke parameters
Pixel and Screen layout	Pixel layout information
Tile fill color	Color for FILL tiling mode
Clear color	Color for fast clear
Filter Parameters	Image filtering parameters
Paint	Paint definitions
Mask	Coverage mask and enable/disable
Error	Oldest unreported error code

Context Parameters [5.2.1]

blue text *is default value*

VGParamType (DataType)
VG_MATRIX_MODE (VGMatrixMode)
- VG_MATRIX_PATH_USER_TO_SURFACE
- VG_MATRIX_PATH_USER_TO_SURFACE
- VG_MATRIX_IMAGE_USER_TO_SURFACE
- VG_MATRIX_FILL_PAINT_TO_USER
- VG_MATRIX_STROKE_PAINT_TO_USER
- VG_MATRIX_GLYPH_USER_TO_SURFACE
- *All Matrix are Initialized to identity matrix*

VG_FILL_RULE (VGFillRule)
- VG_EVEN_ODD
- VG_NON_ZERO

VG_IMAGE_QUALITY (VGImageQuality)
- VG_IMAGE_QUALITY_NONANTIALIASED
- VG_IMAGE_QUALITY_FASTER
- VG_IMAGE_QUALITY_BETTER

VG_IMAGE_MODE (VGImageMode)
- VG_DRAW_IMAGE_NORMAL
- VG_DRAW_IMAGE_MULTIPLY
- VG_DRAW_IMAGE_STENCIL

VG_COLOR_TRANSFORM (VGboolean)
- VG_TRUE
- VG_FALSE

VG_COLOR_TRANSFORM_VLAUES (VGfloat[8])
- { 1.0f, 1.0f, 1.0f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f }
- {Rf, Gf, Bf, Af, Rb, Gb, Bb, Ab}

VG_TILE_FILL_COLOR (VGfloat[4])
- {0.0f, 0.0f, 0.0f, 0.0f} // {R,G,B,A}

VG_CLEAR_COLOR (VGfloat[4])
- {0.0f, 0.0f, 0.0f, 0.0f} // {R,G,B,A}

VG_GLYPH_ORIGIN (VGfloat[2])
- {0.0f, 0.0f} // {ox, oy}

VG_STROKE_LINE_WIDTH (VGfloat)
- 1.0f

VG_STROKE_CAP_STYLE (VGCapStyle)
- VG_CAP_BUTT
- VG_CAP_ROUND
- VG_CAP_SQUARE

VG_STROKE_JOIN_STYLE (VGJoinStyle)
- VG_JOIN_MITER
- VG_JOIN_ROUND
- VG_JOIN_BEVEL

VG_STROKE_MITER_LIMIT (VGfloat)
- 4.0f

VG_STROKE_DASH_PATTERN (VGfloat *)
- { } (array of length 0) (disabled)
- {on1, off1, on2, off2,....}

VG_STROKE_DASH_PHASE (VGfloat)
- 0.0f

VG_STROKE_DASH_PHASE_RESET (VGboolean)
- VG_FALSE
- VG_TRUE

VG_MASKING (VGboolean)
- VG_TRUE
- VG_FALSE(disabled)

VG_SCISSORING (VGboolean)
- VG_TRUE
- VG_FALSE(disabled)

VG_SCISSOR_RECTS (VGint *)
- { } (array of length 0)
- {sx1,sy1,w1,h1,...}

VG_SCREEN_LAYOUT (VGPixelLayout)
VG_PIXEL_LAYOUT (VGPixelLayout)
- VG_PIXEL_LAYOUT_UNKNOWN
- VG_PIXEL_LAYOUT_RGB_VERTICAL
- VG_PIXEL_LAYOUT_BGR_VERTICAL
- VG_PIXEL_LAYOUT_RGB_HORIZONTAL
- VG_PIXEL_LAYOUT_BGR_HORIZONTAL

VG_FILTER_FORMAT_LINEAR (VGboolean)
- VG_TRUE
- VG_FALSE(disabled)

VG_FILTER_FORMAT_PREMULTIPLIED (VGboolean)
- VG_TRUE
- VG_FALSE(disabled)

VG_FILTER_CHANNEL_MASK (VGbitfield)
- (VG_RED | VG_GREEN | VG_BLUE | VG_ALPHA)

VG_BLEND_MODE (VGBlendMode)
- VG_BLEND_SRC
- VG_BLEND_SRC_OVER
- VG_BLEND_DST_OVER
- VG_BLEND_SRC_IN
- VG_BLEND_DST_IN
- VG_BLEND_MULTIPLY
- VG_BLEND_SCREEN
- VG_BLEND_DARKEN
- VG_BLEND_LIGHTEN
- VG_BLEND_ADDITIVE

VG_RENDERING_QUALITY (VGRenderingQuality)
- VG_RENDERING_QUALITY_NONANTIALIASED
- VG_RENDERING_QUALITY_FASTER
- VG_RENDERING_QUALITY_BETTER

Read-Only Context Parameter [5.2.1]

VG_MAX_SCISSOR_RECTS (VGint)
VG_MAX_DASH_COUNT (VGint)
VG_MAX_KERNEL_SIZE (VGint)
VG_MAX_SEPARABLE_KERNEL_SIZE (VGint)
VG_MAX_GAUSSIAN_STD_DEVIATION (VGint)
VG_MAX_COLOR_RAMP_STOPS (VGint)
VG_MAX_IMAGE_WIDTH (VGint)
VG_MAX_IMAGE_HEIGHT(VGint)
VG_MAX_IMAGE_PIXELS (VGint)
VG_MAX_IMAGE_BYTES (VGint)
VG_MAX_FLOAT (VGfloat)

OpenVG ™ API 1.1 Quick Reference Card: Drawing Context

Following is a quick reference to the subset of the OpenVG API specification that pertains to drawing context. **[n.n.n]** refer to the section in the full specification, which is avaiable at www.khronos.org/openvg.

Context Parameter Set/Get API [5.2]

V is void, F is VGfloat, N is VGint

V **vgSeti** (VGParamType *pType*, N *val*)
V **vgSetfv**(VGParamType *pType*, N *cnt*, const F * *val*)
V **vgSetiv**(VGParamType *pType*, N *cnt*, const N * *val*)
F **vgGetf** (VGParamType *pType*)
N **vgGeti** (VGParamType *pType*)
N **vgGetVectorSize**(VGParamType *pType*)
V **vgGetfv**(VGParamType *pType*, N *cnt*, F * *val*)
V **vgGetiv**(VGParamType *pType*, N *cnt*, N * *val*)

Color Transform[13.1]

VG_COLOR_TRANSFORM (VGboolean)
  VG_TRUE
  VG_FALSE
VG_COLOR_TRANSFORM_VLAUES (VGfloat[8])
  { 1.0f, 1.0f, 1.0f, 1.0f, 0.0f, 0.0f, 0.0f, 0.0f }
Example
```
// Lightent : from (0.0~1.0) to (0.7~1.0)
VGfloat trans[8] = { 0.3, 0.3, 0.3, 0.3, 0.7, 0.7, 0.7, 0.7};
vgSeti(VG_COLOR_TRANSFORM , VG_TRUE);
vgSetfv(VG_COLOR_TRANSFORM_VLAUES ,8,  trans);
```

Scissoring [7.1]

VG_COLOR_SCISSORING (VGboolean)
  VG_TRUE
  VG_FALSE
VG_SCISSOR_RECTS (VGint *)
Exmaple
```
#define NUM_RECTS 2
VGint coords[4*NUM_RECTS]
    = { 20, 30, 100, 200, 50, 70, 80, 80 };
// order of x, y, w, h
vgSetiv (VG_SCISSOR_RECTS, 4*NUM_RECTS, coords);
```

Quering Hardware Capabilities [14]

VGHardwareQueryType
  VG_IMAGE_FORMAT_QUERY
  VG_PATH_DATATYPE_QUERY
VGHardwareQueryResult
  VG_HARDWARE_ACCELERATED
  VG_HARDWARE_UNACCELERATED
Query Command
VGHardwareQueryResult **vgHardwareQuery**
  (VGHardwareQueryType *key*, N *setting*)

Masking [7.2]

V is void, F is VGfloat, N is VGint, M is VGMaskLayer

M **vgCreateMaskLayer** (N *w*, N *h*)
V **vgDestroyMaskLayer**(M *mask*)
V **vgMask**(H mask, VGMaskOperation *op*,
  N *x*, N *y*, N *w*, N *h*)
V **vgRenderToMask** (VGPath *p*,
  VGbitfield *paintMode*, VGMaskOperation *op*)
  // VG_STROKE_PATH and or VG_FILL_PATH
V **vgFillMaskLayer**(M *mask*, N *x*, N *y*, N *w*, N *h*, F *val*)
V **vgCopyMask**(M *mask*, N *x*, N *y*, N sx, N sy,
  N *w*, N *h*, F *val*)

VGMaskOperation
```
// M is new mask, Mn is input mask,
// Mp is prev. Mask
    VG_CLEAR_MASK       M = 0
    VG_FILL_MASK        M = 1
    VG_SET_MASK         M = Mn
    VG_UNION_MASK       M = 1 - (1-Mn)*(1-Mp)
    VG_INTERSECT_MASK   M = Mn * Mp
    VG_SUBTRACT_MASK    M = Mp * (1-Mn)
```

Transform [6]

Matrix m = { sx, shy, w0, shx, sy, w1, tx, ty, w2 }
in Affine Transform w0 = w1 = 0.0, w2 = 1.0

Select Matrix Mode

vgSeti(VG_MATRIX_MODE , VGMatrixMode)
parameter VG_MATRIX_MODE (VGMatrixMode)

VG_MATRIX_PATH_USER_TO_SURFACE	(Affine)
VG_MATRIX_PATH_USER_TO_SURFACE	(Affine)
VG_MATRIX_IMAGE_USER_TO_SURFACE	(Perspective)
VG_MATRIX_FILL_PAINT_TO_USER	(Affine)
VG_MATRIX_STROKE_PAINT_TO_USER	(Affine)
VG_MATRIX_GLYPH_USER_TO_SURFACE	(Affine)

Transform Function [6]

V is void, F is VGfloat

V **vgLoadIdentity** ( )
V **vgLoadMatrix** (const F* *m*)
V **vgMultMatrix** (const F* *m*)
V **vgGetMatrix** (F* *m*)
V **vgTranslate** (F *tx,* F *ty*)
V **vgScale** (F *sx,* F *sy*)
V **vgShear** (F *shx,* F *shy*)
V **vgRotate** (F *angle*)

Blending Equation[13.2]

alpha blending function $\alpha(\alpha_{src}, \alpha_{dst})$ / color blending function $c(c_{src}, c_{dst}, \alpha_{src}, \alpha_{dst})$
pre-mult alpha form $c'(\alpha_{src} * c_{src}, \alpha_{dst} * c_{dst}, \alpha_{src}, \alpha_{dst}) = c'(c'_{src}, c'_{dst}, \alpha_{src}, \alpha_{dst})$

Blend Mode	$c'(c'_{src}, c'_{dst}, \alpha_{src}, \alpha_{dst})$	$\alpha(\alpha_{src}, \alpha_{dst})$
Porter-Duff Blending		
VG_BLEND_SRC	c'_{src}	α_{src}
VG_BLEND_SRC_OVER	$c'_{src} + c'_{dst} * (1-\alpha_{src})$	$\alpha_{src} + \alpha_{dst} * (1-\alpha_{src})$
VG_BLEND_DST_OVER	$c'_{src} * (1-\alpha_{dst}) + c'_{dst}$	$\alpha_{src} * (1-\alpha_{dst}) + \alpha_{dst}$
VG_BLEND_SRC_IN	$c'_{src} * \alpha_{dst}$	$\alpha_{src} * \alpha_{dst}$
VG_BLEND_DST_IN	$c'_{dst} * \alpha_{src}$	$\alpha_{dst} * \alpha_{src}$
Additional Blending		
VG_BLEND_MULTIPLY	$c'_{src} * (1-\alpha_{dst}) + c'_{dst} * (1-\alpha_{src}) + c'_{src} * c'_{dst}$	$\alpha_{src} + \alpha_{dst} * (1-\alpha_{src})$
VG_BLEND_SCREEN	$c'_{src} + c'_{dst} - c'_{src} * c'_{dst}$	$\alpha_{src} + \alpha_{dst} * (1-\alpha_{src})$
VG_BLEND_DARKEN	$\min(c'_{src} + c'_{dst} * (1-\alpha_{src}),\ c'_{dst} + c'_{src} * (1-\alpha_{dst}))$	$\alpha_{src} + \alpha_{dst} * (1-\alpha_{src})$
VG_BLEND_LIGHTEN	$\max(c'_{src} + c'_{dst} * (1-\alpha_{src}),\ c'_{dst} + c'_{src} * (1-\alpha_{dst}))$	$\alpha_{src} + \alpha_{dst} * (1-\alpha_{src})$
Additive Blending		
VG_BLEND_ADDTIVE	$\min(c'_{src} + c'_{dst}, 1)$	$\min(\alpha_{src} + \alpha_{dst}, 1)$

Stencil Equation [10.8]

in stencil mode, alpha and color equation for blending is changed to

$R_{dst} \leftarrow c(R_{paint}, R_{dst}, R_{image} * \alpha_{image} * \alpha_{paint}, \alpha_{dst}) / \alpha_{tmp}$
$G_{dst} \leftarrow c(G_{paint}, G_{dst}, G_{image} * \alpha_{image} * \alpha_{paint}, \alpha_{dst}) / \alpha_{tmp}$
$B_{dst} \leftarrow c(B_{paint}, B_{dst}, B_{image} * \alpha_{image} * \alpha_{paint}, \alpha_{dst}) / \alpha_{tmp}$
$\alpha_{dst} \leftarrow \alpha_{tmp}$

$\alpha_{tmp} = \alpha(\alpha_{image} * \alpha_{paint}, \alpha_{dst})$
$L_{dst} \leftarrow c(L_{paint}, L_{dst}, L_{image} * \alpha_{image} * \alpha_{paint}, \alpha_{dst}) / \alpha_{tmp}$
$\alpha_{dst} \leftarrow \alpha_{tmp}$
// drawing surface has a luminance-only format