August 2010

OpenVG Graphics Solutions Tutorial
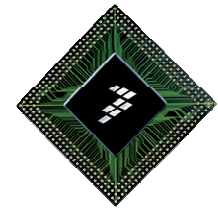
Huang Loren

MAD Senior Software Engineer / R&D

freescale ™
semiconductor

▶ Problems in 2D Graphic

- No existing standard API
- Graphic resolution dependant

▶ Vector Graphic - OpenVG

- Industry standard by Khronos Consortium
- Fulfills requirements of key applications

freescale ™
semiconductor

Session Objectives

▶ After completing this session you will:

- Understand the unique features of Vector graphics

- Have an overview of the OpenVG rendering pipeline

- Be familiar with the basic groups of functionality in OpenVG

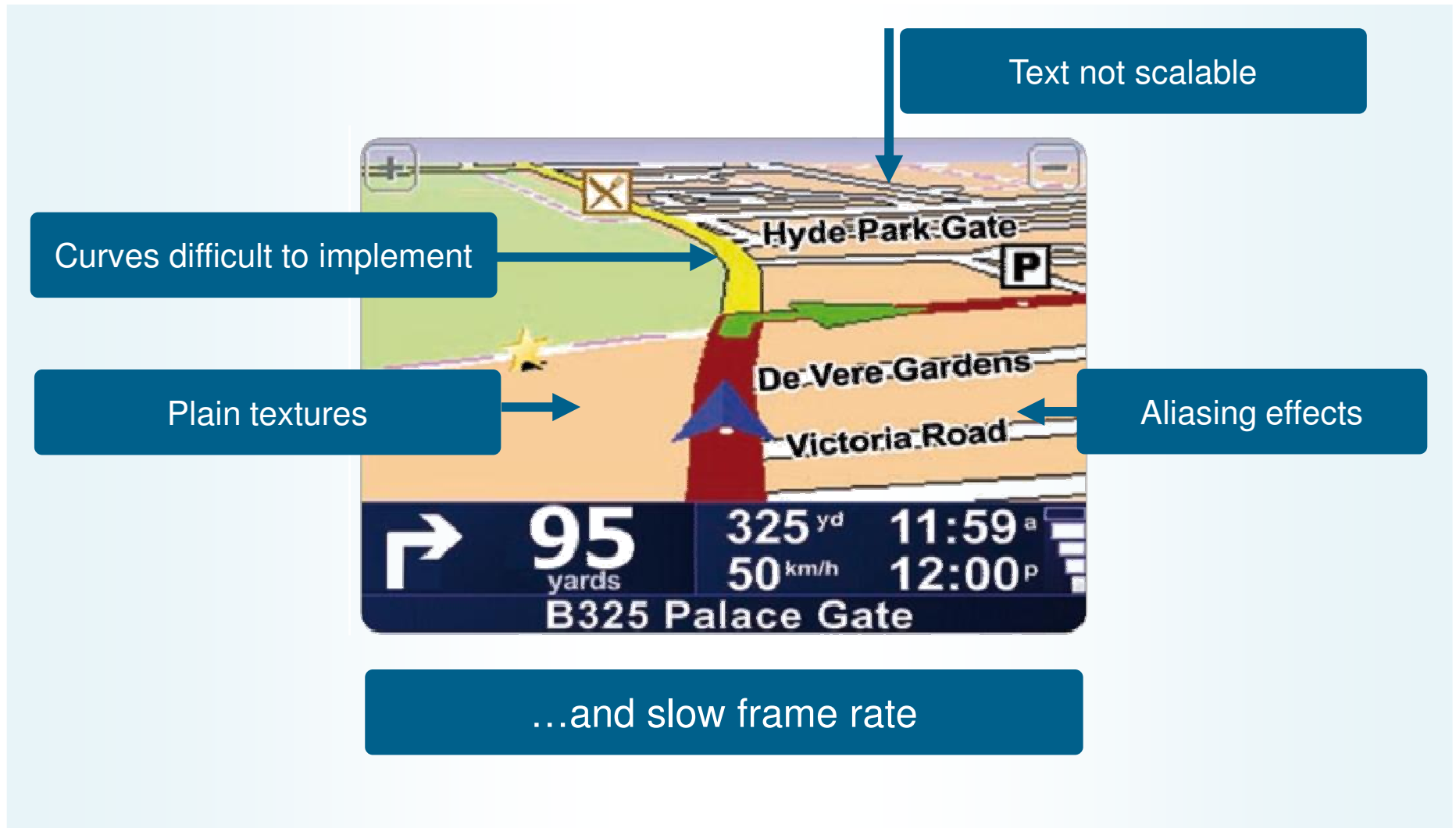- Understand how to port graphics to different processors

freescale ™
semiconductor
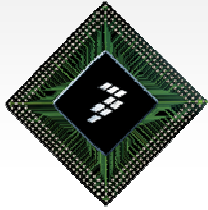
Agenda

►**OpenVG at a Glance**

►**OpenVG**

- Rendering Pipeline

- Programming Model

Problems with Existing Graphics

Text not scalable

Curves difficult to implement

Plain textures

Aliasing effects

Hyde Park Gate

P

De Vere Gardens

Victoria Road

95 yards

325 yd   11:59 a
50 km/h   12:00 p

B325 Palace Gate

...and slow frame rate

freescale ™
semiconductor

What Are Vector Graphics?

► **Vector graphics** are drawn and stored as mathematical vector formulae

► Each vector and fill is assigned color value, instead of assigning color to each separate pixel

► A black circle can be represented as:

- **$x = r \cos \theta$**
  $y = r \sin \theta$

or:

- **$x^2 + y^2 = r^2$**

- *With color value 0000 for black*

► **Benefits**
- Infinitely zoomable
- Independent of screen resolution
- Saves data memory

freescale ™
semiconductor
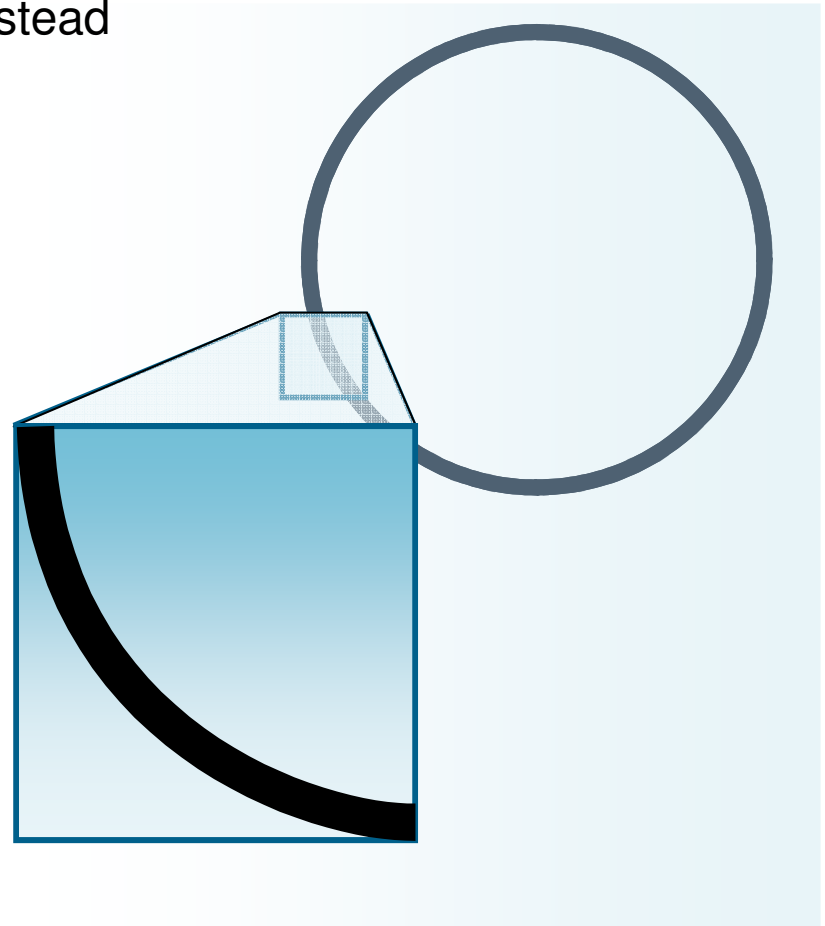
Where Are Vector Graphics Used?

► **Graphic Design**

  • e.g., Adobe™ Photoshop, Illustrator, Flash

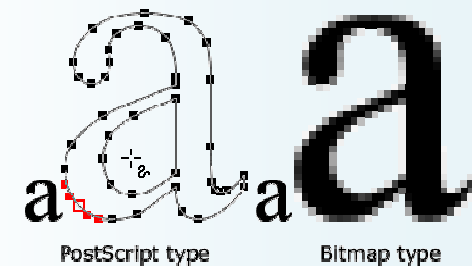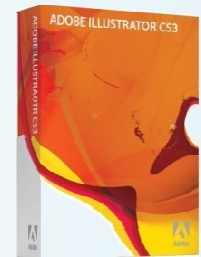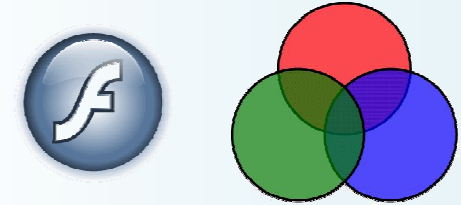► **Vector Graphics Languages**

  • e.g., SVG, VML, Postscript

► **World Wide Web**

  • Most web content uses vector graphics
 • Most web browsers render SVG content directly

► **Vector Graphics Libraries**

  • e.g., Cairo
    ▪ used by GTK+, Mozilla, Webkit, etc.

PostScript type   Bitmap type

freescale ™
semiconductor

► Design Philosophy

- Expands the OpenGL programming model to 2D vector graphics
- Provides a low-level hardware acceleration abstraction layer
- Uses OpenGL-style syntax where possible
- Allows flexibility in the way acceleration can be provided
- Enables hardware vendors to use their own preferred internal representations
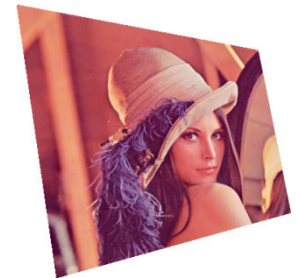
► The VGU Utility Library

- Higher-level Geometric Primitives
- Image Warping
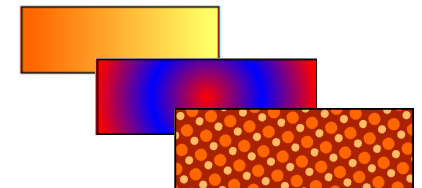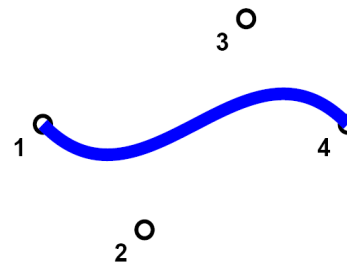
(Some) OpenVG highlights

Path (Open path; Closed path)

The path is the basis for all vector objects. A path is made up of one or more line segments connected by two or more anchor points. Paths can be made from a combination of straight lines and curves, each of which may be made up of many connecting points. Paths can be open or closed. An open path is one with unconnected end points, while a closed path is one whose start and end points meet.
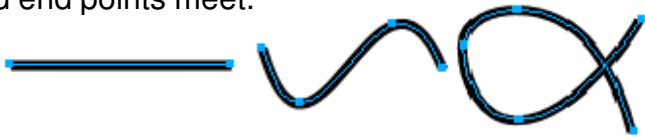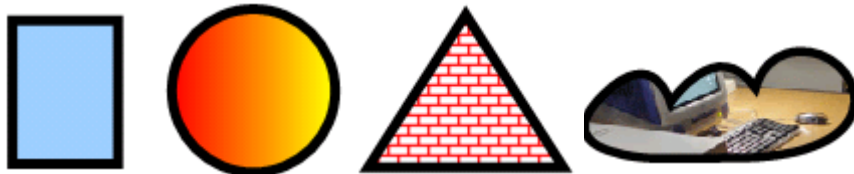
Open paths (anchor points and line segments shown in blue)

Closed paths (anchor points and line segments shown in blue)

Fill

A fill can be applied to any area within a path. Fills can be single blocks of colour, gradients, patterns or images (raster or vector).

Coloured fill   Gradient fill   Patterned fill

Bézier curve (Curve)

A curved segment of a path is known as a Bézier curve (after French mathematician Pierre Bézier). Bézier curves are defined by mathematical equations - essentially, the coordinates of a curve can be calculated and drawn by knowing the position of two end points and two control points.

Control points

End points

freescale ™
semiconductor
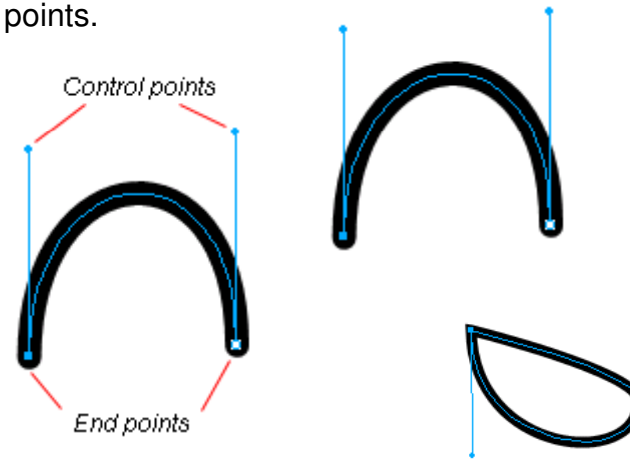
OpenVG Applications

► **Acceleration of Existing Content**

- Accelerates Flash Lite, SVG Tiny, potentially Silverlight

► **Great for GUI Acceleration**

- Dynamic Vector Font Rendering
- Provides variety of advanced blending and drawing functions

► **Great for Accelerating User Applications**

- Fast, power-efficient, scalable, skinnable user-interfaces including animations and different aspect ratios
- Web browsing
- Navigation applications
  - Accurate map rendering at interactive frame rates
- Cartoons, anime, greeting cards, games, mobile entertainment

freescale ™
semiconductor
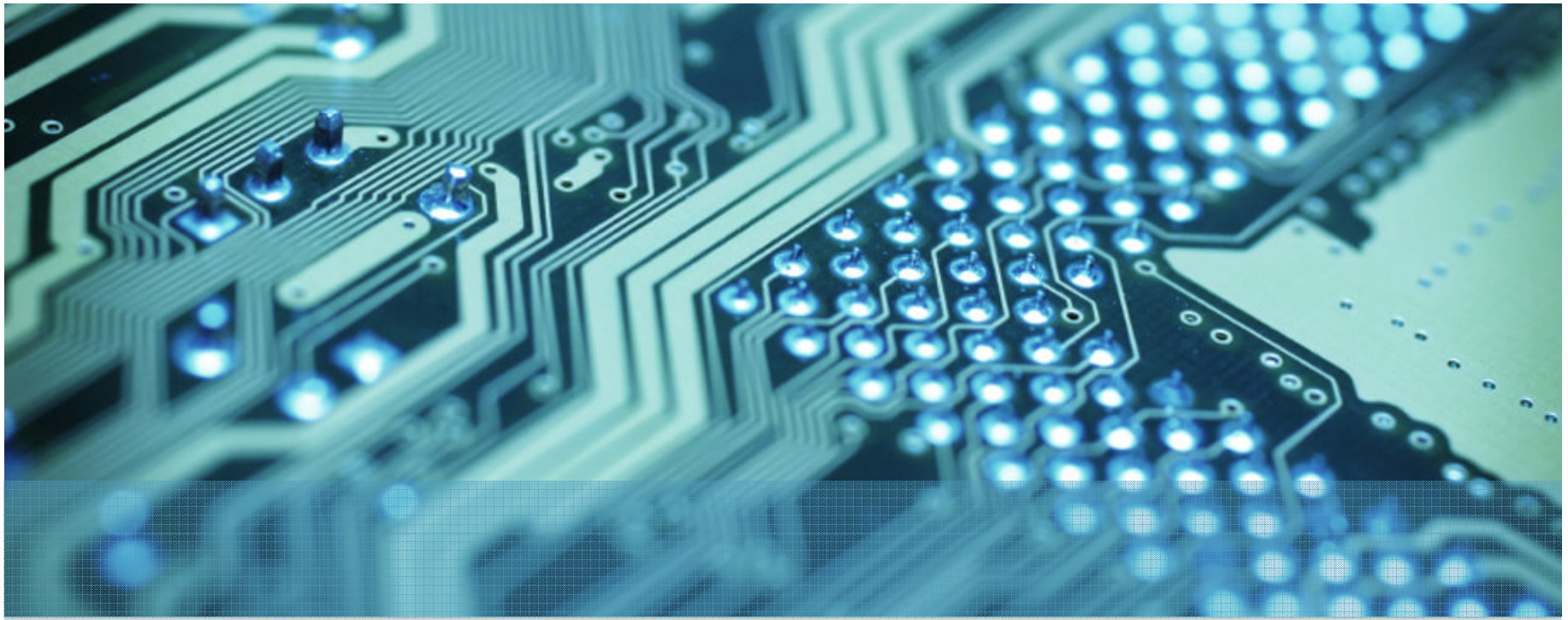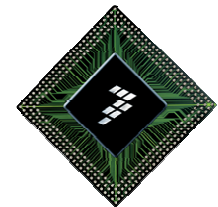
OpenVG Rendering Pipeline

► OpenVG defines a hardware pipeline for paths and images

- Path Definition
- Stroking
  - Line width, joins & caps, dashing, etc.
- Transformation
  - 2 x 3 (paths) and 3 x 3 (images) transformations
- Rasterization
- Clipping & Masking
  - Scissor rectangles
- Paint Generation
  - Flat color, gradient, or pattern paint
- Blending
  - Multiple blend modes
- Dithering
- Image Filters

Path Definition & Setting of API Parameters

Stroking

Transformation

**Rasterization**

Clipping and Masking

Paint Generation

Blending

Dithering

freescale ™
semiconductor

▶ MOVE_TO, LINE_TO, QUAD_TO, CUBIC_TO, CLOSE_PATH

- Elliptical Arcs
- Absolute / Relative Coordinates
- Smooth Curves
- Path Interpolation
- Path queries:
  - Bounding boxes
  - Point along path
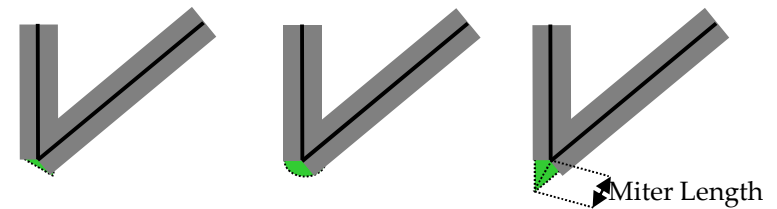 - Tangent along path
- Non-Zero and Even-Odd fill rules

(px, py)

(sx, sy)

MOVE_TO

LINE_TO

CUBIC_TO

(ox, oy)

freescale ™
semiconductor
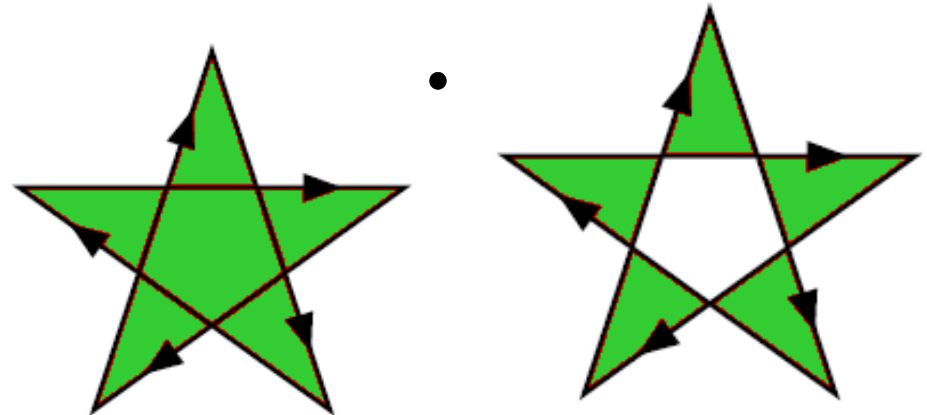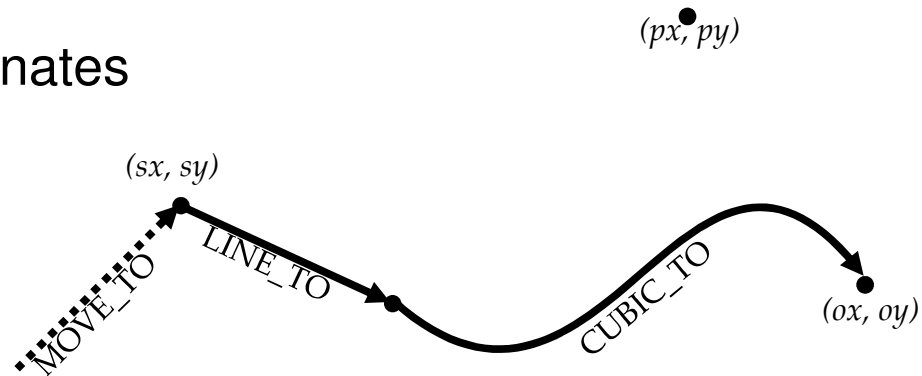
► Stroking takes a path and defines an outline around it based on:

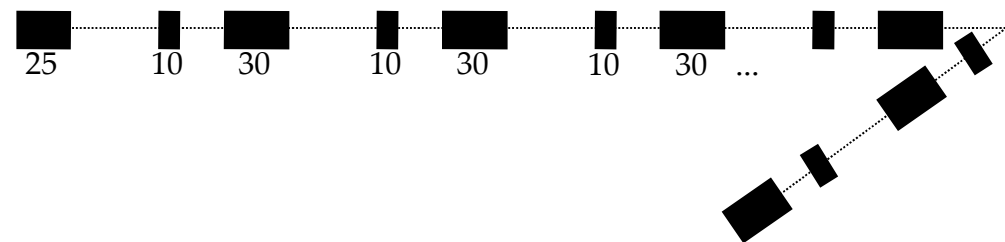- Line width
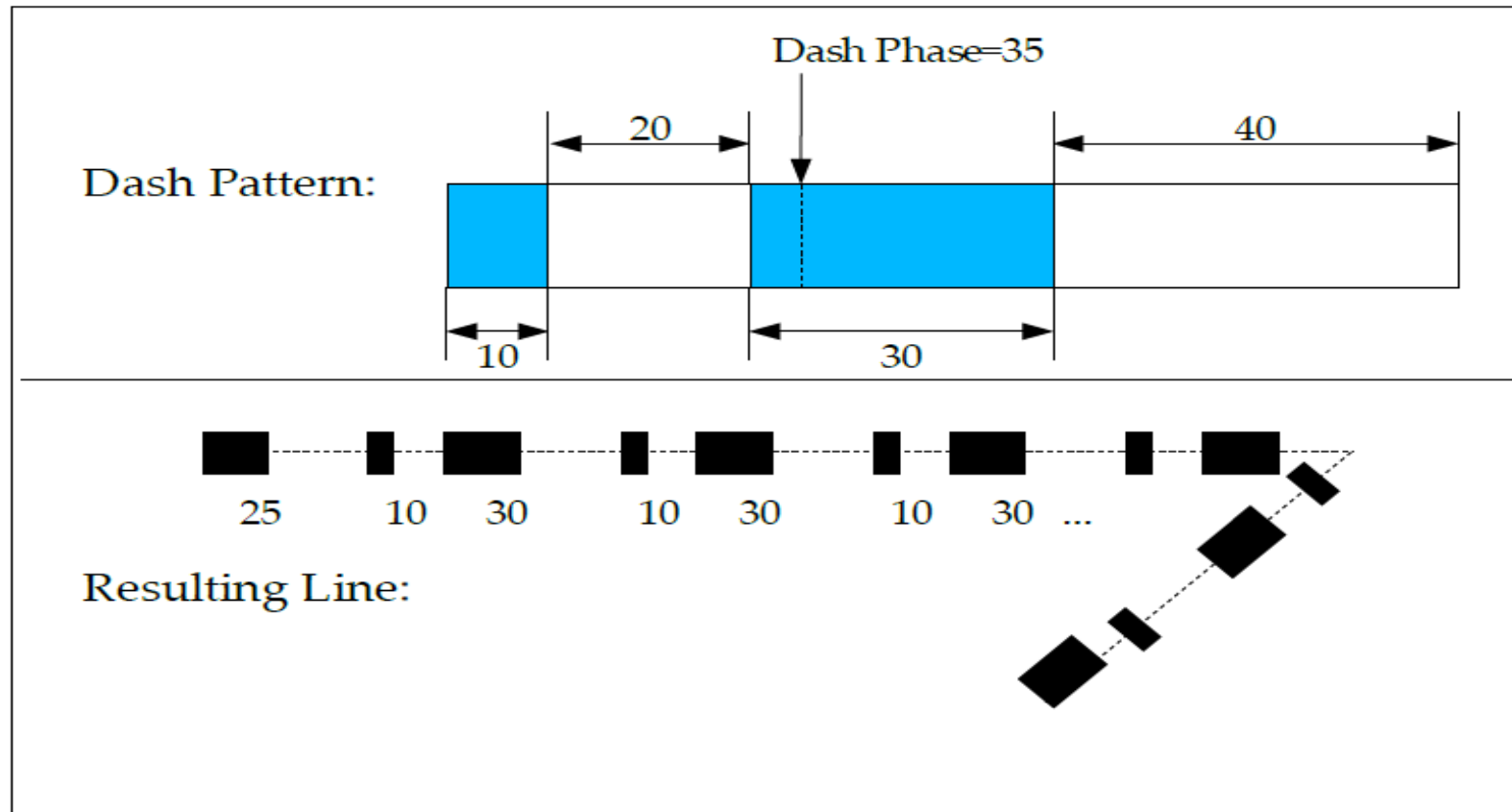- End cap style (butt, round or square)

- Line join style (bevel, round or miter)

Miter Length

- Miter limit (to convert long miters to bevels)
- Dash array and offset
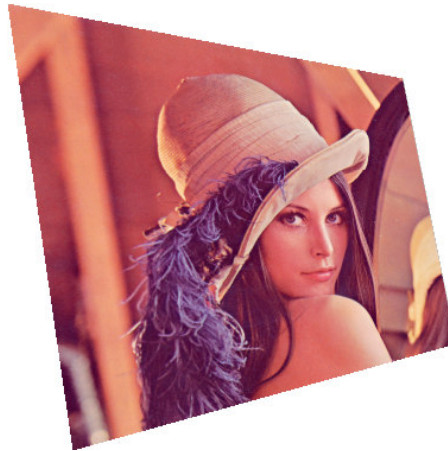
Dash array = { 10, 20, 30, 40} / Dash Phase = 35

25 10 30 10 30 10 30 ...

freescale ™
semiconductor

Dash array = { 10, 20, 30, 40} / Dash Phase = 35

freescale ™
semiconductor

▶ Paths use 2 x 3 affine transformations

▶ Images use 3 x 3 perspective transformations

▶ Transformation functions are similar to OpenGL:

- vgLoadIdentity
- vgLoadMatrix
- vgGetMatrix
- vgMultMatrix
- vgScale
- vgRotate
- vgTranslate

$$\begin{bmatrix} 1.080 & 0.101 & 0 \\ 0.209 & 0.691 & 0 \\ 1.28 \times 10^{-3} & -1.19 \times 10^{-3} & 1 \end{bmatrix}$$

▶ NOTE: If you want to transform a VG path / object into a perspective, you must do that before sending the points to OpenVG

freescale ™
semiconductor

► The goal of rasterization is to determine a filtered alpha value for each pixel, based on the geometry around that pixel

► Filters may be up to 3 pixels in diameter

► OpenVG handles rasterizing arbitrary shapes

→
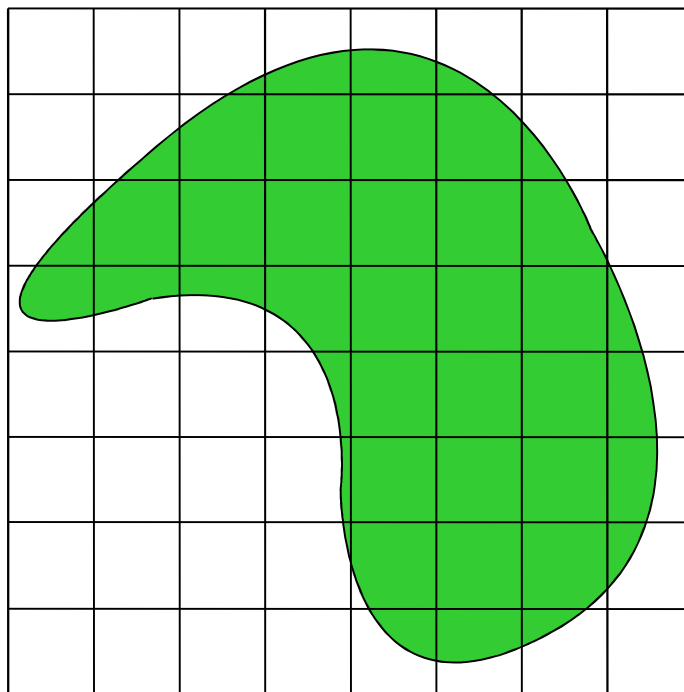
0	0	.1	.4	.5	.2	0	0
0	.3	.8	1	1	.9	.4	0
.4	.8	1	1	1	1	.8	0
.6	.4	.3	.7	1	1	1	.3
0	0	0	.2	1	1	1	.5
0	0	0	.1	1	1	1	.5
0	0	0	.1	.9	1	.9	.2
0	0	0	0	.3	.5	.2	0

freescale ™
semiconductor

► Pixels outside a rectangular viewport are not drawn.

► Only pixels inside a set of scissor rectangles are drawn.

► Clip-rects help to cut down the amount of pixels rendered in cases that have only a portion of the screen being rendered. This is essential for GUIs and can help with both maximizing performance and minimizing power.

Viewport

Scissor Rectangles

Pixels drawn

Pixels not drawn

freescale ™
semiconductor
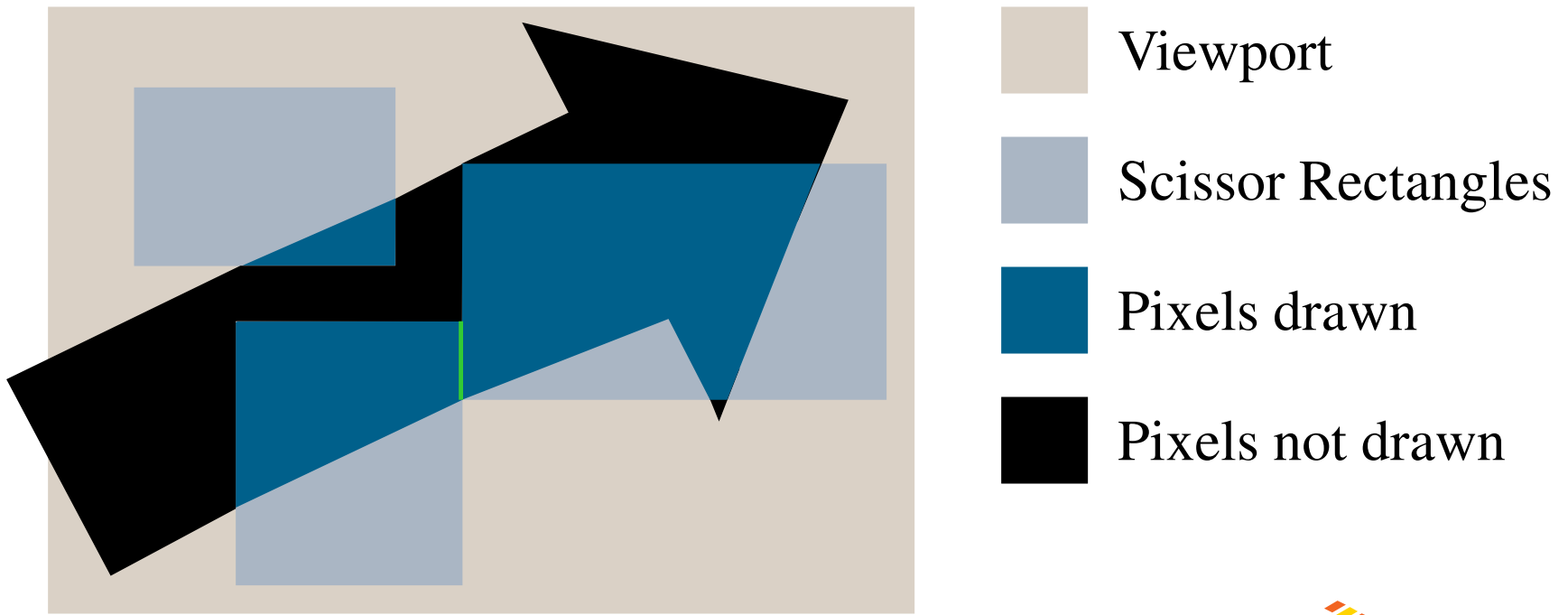
► In addition to clipping, a per-pixel mask may be applied

► The mask has an alpha value at each pixel that is multiplied by the alpha from the rendering stage

► May be used to "cut out" an area, create a transition between areas

► Mask values may be modified using image data
- Fill, Clear, Set, Add, Subtract, Intersect

Generating a Mask is a 'draw' action and relatively expensive. Masks should be generated infrequently to minimize performance impact.

freescale ™
semiconductor

Paint Generation

▶ Paint is generated pixel-by-pixel and applied on top of geometry

▶ The alpha from the previous stage (rendering + masking) is used to determine how much paint to apply

▶ Separate paint objects for stroking, filling

▶ Paint is applied through an affine transformation

▶ Four types of paint are supported:

  · Flat color paint

  · Linear Gradient paint: Points (x1, y1) and (x2, y2), color ramp

  · Radial Gradient paint: center (x, y), focus (x, y), radius, color ramp

  · Pattern paint based on an image, tiling mode

Increasing Performance Cost

(x1, y1)

(x2, y2)

r

(cx, cy)

freescale ™
semiconductor
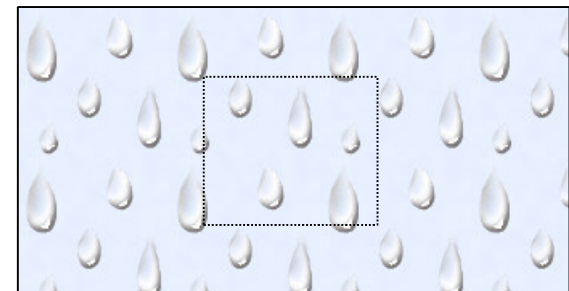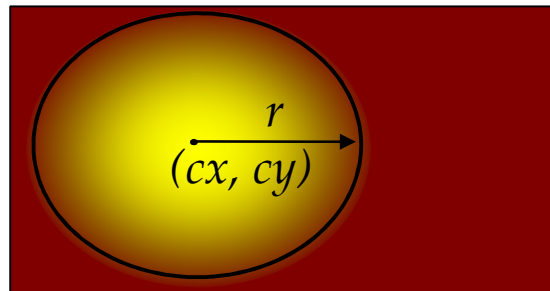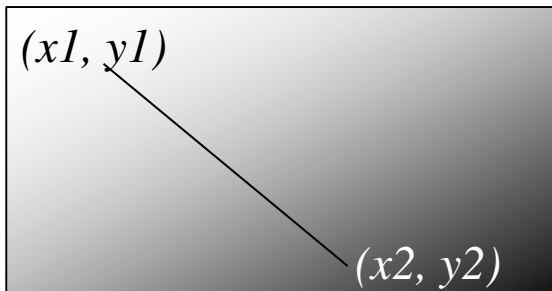
Blending

▶ Combine masked alpha from path with paint alpha

▶ Blend the result onto the drawing surface

▶ Blending is a function of:

- The paint (R, G, B) color
- The masked alpha value (path alpha   mask alpha   paint alpha)
- The destination (R, G, B) color
- The destination alpha value (1 if no stored alpha)

▶ There are 8 blending functions:

- Porter-Duff "source" mode (copy source to destination)
- Porter-Duff "source over destination"/ "destination over source"
- Porter-Duff "source in destination"/ "destination in source"
- Lighten (choose lighter of source and destination)
- Darken (choose darker of source and destination)
- Multiply (black source pixel forces black, white leaves unchanged)
- "Screen"(white source pixel forces white, black leaves unchanged)
- Additive (add pixel values, add alpha up to 1)

▶ Blending may occur in a linear or non-linear color space

freescale ™
semiconductor

► Images are defined using one of 15 pixel formats
- Linear or non-linear (sRGB) color spaces
- Linear or non-linear grayscale
- Pre-multiplied or non-premultiplied alpha
- 8/8/8, 5/6/5 or 5/5/5 bit depths
- 1-bit Black & White (e.g., for Fax applications)

► Images may be stored in accelerated memory

► Image filters may be applied:
- Color Matrix
- Convolve, Separable Convolve, Gaussian Blur
- Lookup, LookupSingle
- Dither

► Images may be drawn in perspective

► Image may be used as a stencil to apply paint
- Very useful for drawing anti-aliased text

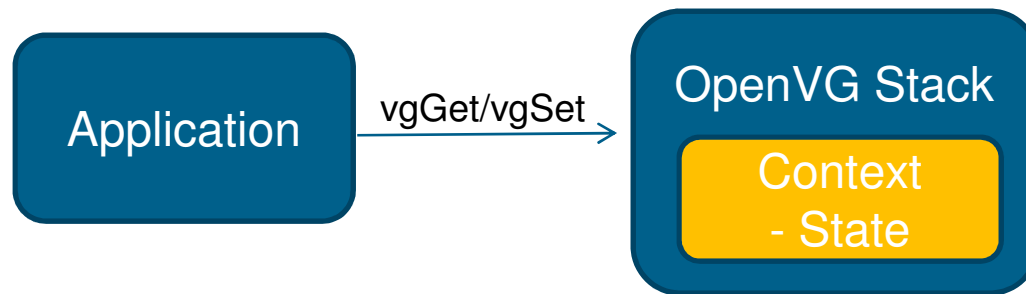*freescale* ™
semiconductor
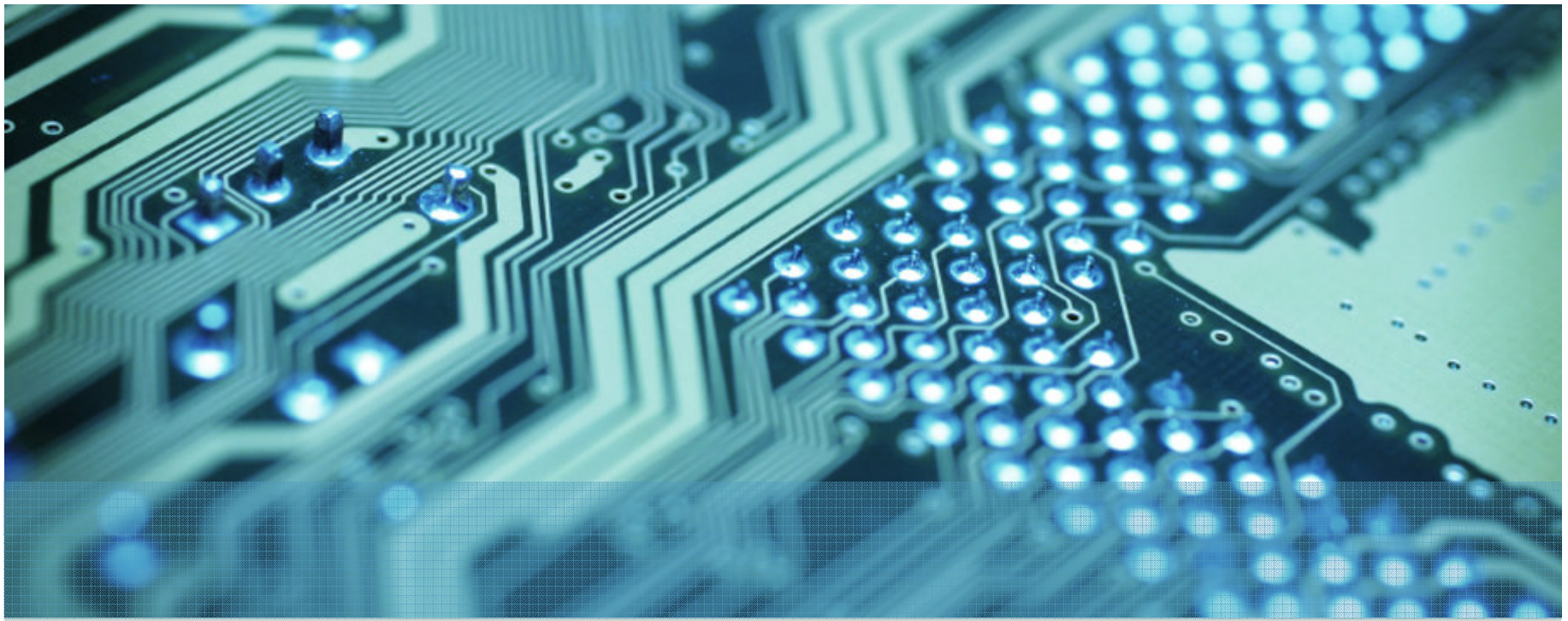
OpenVG Programming Model
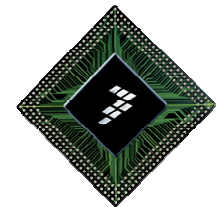
OpenVG Programming Model

▶ OpenVG is designed around a State Machine based client-server model

- User 'sets' and 'gets' variables in the machine (enable/disable, bind, etc.)

Application	vgGet/vgSet →	OpenVG Stack
		Context - State

- OpenVG: handles to paints, paths, gradients, etc., to avoid the re-preprocessing of each frame

▶ Data types in a function are determined by a i, f, or v postfix

- Integer, float, or vector

▶ Execution decoupled

- Execution is only guaranteed when user blocks (vgFinish)
- Or flips the drawing surfaces (eglSwapBuffers)

freescale ™
semiconductor

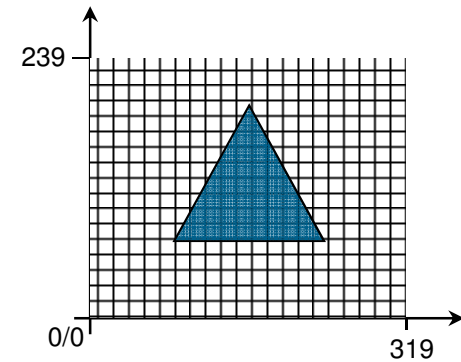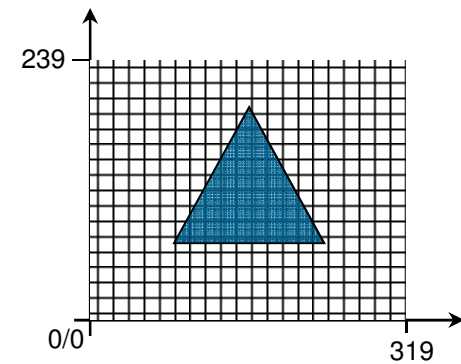# Coordinate System

User Coordinates
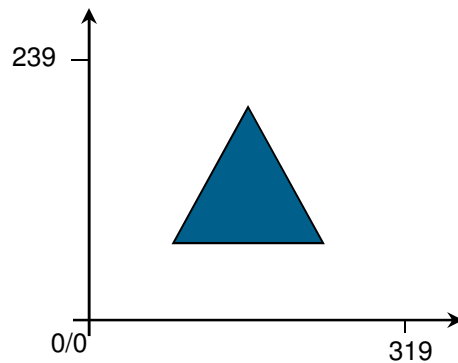
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Screen Coordinates
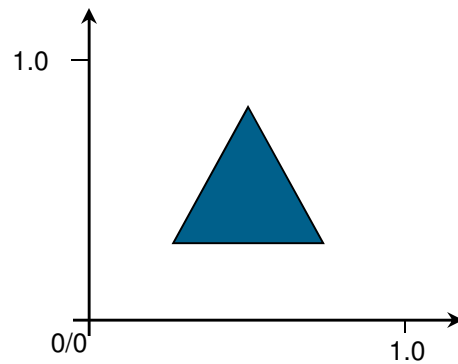
$$\begin{bmatrix} 319 & 0 & 0 \\ 0 & 239 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

freescale ™
semiconductor

OpenVG: Documentation

▶ The Reference is the OpenVG Standard 1.1 as published by the Khronos Consortium.

   It is available from their website: http://www.khronos.org/openvg

freescale ™
semiconductor

freescale™

semiconductor