



# ZOOM OUT BREW 2008 CONFERENCE

Using Scalable Vector Graphics (SVG) for  
Better BREW Apps

James Hicks, Staff Engineer, Qualcomm Incorporated

Michael Curtes, Sr. Project Manager, Faith West Inc.

# Agenda

- SVG Overview
- BREW and SVG Overview
- Current market status
- Creating SVG content for BREW applications
- Enabling SVG interactivity through BREW application
- Optimizing SVG for different device tiers
- SVG roadmap

Note that this presentation will be available for download from the BREW Conference 2008 website. Additional information is available in the *SVG-CMX Developer Guide* distributed at today's presentation.

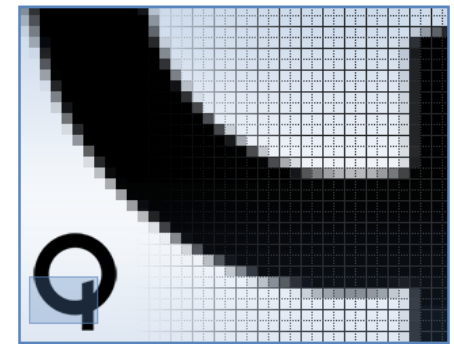
# Scalable Vector Graphics (SVG) Overview

- 2D vector graphics platform
- Standardized by the W3C and 3GPP
- Two parts:
  - XML-based file format
  - Programming API for graphical applications
- Key Features: Shapes, text and embedded raster graphics
- Supports embedded scripting through languages like ECMAScript and JavaScript
- Comprehensive support for animation
- Support for embedded audio and video
- Used to create rich static and animated content for:
  - The Web
  - Mobile phones
  - Car navigation

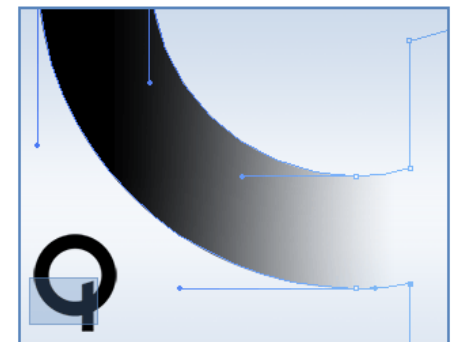


# Benefits of SVG

- Content is scalable across varying handset display sizes
- Smaller file sizes than rasterized formats (bmp, png, gif, jpg)
- Standards-based
- More than just image format
  - Interactivity
  - Integrate other rich media content
  - Text is searchable and dynamic
    - Not just a graphic
    - Replace specific text with server data
- Modular designer/developer workflow
- Solid, easy-to-use authoring tools
- XML-based format is easy to create and edit
- Dynamic, server-generated content



Raster Image



Vector Image

# Current SVG in Mobiles Market Support

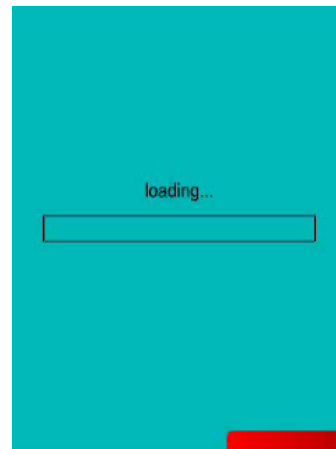
- Over 275 million\* SVG handsets globally
- Over 200 SVG handset models globally
  - Nokia, Sony Ericsson and Sharp are all strong SVG proponents
- Already strong in Europe, the U.S. and Asia
  - SVG implementation required in Japan
    - Central to KDDI's EZNavi (navigation) and EZ Document Viewer
  - European carriers requiring SVG implementation
    - Core technology behind several Vodafone services
  - US Carriers mandating SVG and planning SVG-based content services
- SVG support included in iPhone
  - Within Safari browser
- WebKit includes SVG support
  - Open source web browser engine
  - Used on Android, iPhone and Nokia N Series platforms

\* Device numbers are estimated

# SVG Content Services

Because of its efficiency and flexibility, SVG is used for a wide range of applications and services.

- **Ringtones**
- **Screensavers & Wallpapers**
- **Handset UI**
- **Gaming**
- **GPS Navigational Mapping**
- **Enterprise Applications**
- **Mobile TV**
- **On-device Portals**
- **Comic Applications**
- **Interactive Advertising**
- **Document Viewing**
- **Mobile Web**
- **Mobile advertising**
- **Music distribution**
- **Electronic book or magazine**
- **In MMS**
- **In Browsers**
- **In BREW and J2ME games and applications**



# BREW and SVG – Current Status

- Basic SVG animation playback is supported through **IMedia** interface since BREW 3.1.4
  - Play, pause, stop, rewind, fast forward, etc
- As of BREW 3.1.4, a set of specific SVG-related APIs are supported through **IMediaSVG** interface
  - Key presses, pointer actions, rotate, pan, zoom, focus, etc.
- BREW SDK 3.1.4 and higher include **IMediaSVG** APIs in **AEMediaSVG.h** header file

# BREW and SVG – Current Status continued

- BREW SDK SVG extension available
  - Emulation of SVG Tiny 1.2 player in BREW Simulator supported
  - ‘Apron’ BREW app (including source code)
    - A simple interactive BREW dynamic application designed for basic preview and interaction with SVG content
  - Additional sets of SVG-related APIs
    - **IMEDIASVG** – SVGT 1.2 APIs
    - **ISVGDOM** –  $\mu$ DOM functionality APIs

# BREW and SVG – Current Status continued

- Most recent SVG implementation supports SVG Tiny 1.2 specification
- $\mu$ DOM functionality
  - BREW SVG DOM APIs can be used to insert, remove, modify, and interact with nodes/elements in the SVG content
  - Useful for hard-wiring device soft keys with SVG elements
  - Useful for modifying SVG content on-the-fly, using server data, etc.

# Designer/Developer Workflow

- Designer and developer agree on functionality and design of application
  - Agree on IDs of 'hooks' to be used in connecting SVG and BREW applet code
- Designer creates SVG
  - Uses graphic authoring tool or text editor
- Functionality and interactivity authored in SVG itself
- BREW developer creates code which sends user input to SVG content
- Modifications are made to SVG content when necessary without changing BREW application code

# Creating SVG Content

- Several SVG authoring tools available
  - Oracle *Mobile Designer 2.X*™
  - Ikivo *Animator 2.X*™
  - Adobe® *Illustrator CS3*® or later
  - Adobe® *GoLive CS3*® or later
  - *Inkscape 0.4X* (open source)

ORACLE® | Hyperion®  
**MobileDesigner**



# Creating SVG Content for BREW

- Include XML element IDs on key elements

```
<g id="Button_Back2"> ...
```

- IDs used by BREW developer when hooking into SVG content
- Makes reviewing SVG content by others easy

- Include focus ring settings

```
<g id="Button_Back2" nav-prev="url(#Button_GetNew)" nav-next="url(#Button_Options5)"> ...
```

- Define elements' 'nav-prev' and 'nav-next', etc. to guarantee intended navigation

- Disable focusability when necessary

```
<g id="Animation_38" focusable="false" > ...
```

- Prevents unnecessary clicks when navigating

# Creating SVG Content for BREW continued

- Do not include absolute dimensions

```
<svg width="100%" height="100%"> ...
```

- Allows content to scale to all devices' display sizes

- Consider preserving aspect ratio

```
<svg preserveAspectRatio="none"> ...
```

- "none" ensures the entire display is used, but can cause undesirable skewing on different screen shapes
- "xMidYMid" prevents skewing as above, but may not make full use of the display

- Use gzip compression

- Compresses SVG by up to 80%

# Creating SVG Content for BREW continued

- Optimize performance with the “static” property

```
<g static="true"> ...
```

- Causes that element to be rendered into an off-screen cache buffer
- Apply only to the <g> element
- Do not use the “static” property on items that have frequently-occurring animation
- Avoid grouping together many items that are not near each other onscreen and giving them all the “static” attribute at once
- If in doubt, experiment with the “static” attribute and measure the results

# Applying the “static” Attribute

- DO apply **static** to each individual icon
- DO apply **static** to each of the individual bubble clusters
- DON'T apply **static** to the entire grid of icons as one group
- DON'T apply **static** to a single group enclosing both bubble clusters



# Optimizing SVG for different device tiers

- Software-only SVG
  - Avoid overlapping elements where possible
  - Make careful use of the “static” attribute
  - Pre-render as much complex SVG into embedded bitmaps as practical
- OpenGL ES-accelerated SVG
  - Avoid very complex paths
  - Avoid large-scale zooming
- Native hardware-accelerated SVG
  - No restrictions

# Integrating SVG into BREW Application

- Initial set up required in order to display SVG in BREW app

1. Obtain an **IMediaSVG** interface pointer using **AEECLSID\_MEDIASVG** class ID

```
ISHELL_CreateInstance(pMe->a.m_pIShell, AEECLSID_MEDIASVG,  
    (void**) &pMe->pIMediaSVG);
```

2. Initialize the interface with the content filename to be played

```
AEEMediaData mData;  
mData.clsData = MMD_FILE_NAME;  
mData.pData = svg_filename;  
mData.dwSize = 0;  
IMEDIA_SetMediaData(pMe->pIMediaSVG, &mData);
```

3. Register a callback function to be called when a frame is completed (optional)

```
IMEDIA_RegisterNotify(pMe->pIMediaSVG, MediaNotifyCB, pMe);
```

4. Play the file

```
IMEDIA_Play(pMe->pIMediaSVG);
```

# Integrating SVG into BREW Application continued

- The callback function, if present, is responsible for obtaining a frame and displaying the frame to the LCD (optional)

```
void MediaNotifyCB(void * pUser, AEEMediaCmdNotify * pCmdNotify)
{
    APP_CLASS * pMe = (APP_CLASS*) pUser;
    switch(pCmdNotify->nStatus)
    {
        case MM_STATUS_FRAME:
        {
            AEEDeviceInfo di;
            IBitmap* frameBuf;
            IMEDIA_GetFrame(pMe->pIMediaSVG, &frameBuf);
            ISHELL_GetDeviceInfo(pMe->a.m_pIShell, &di);
            IDISPLAY_BitBlt(pMe->a.m_pIDisplay, 0, 0, di.cxScreen,
                di.cyScreen, frameBuf, 0, 0, AEE_RO_COPY);
            IDISPLAY_UpdateEx(pMe->a.m_pIDisplay, FALSE);
            break;
        }
    }
    return;
}
```

## Integrating SVG into BREW Application continued

- Connecting SVG events to device keys
  - Using focus events:
    - Navigate through the focus ring using the **PREVIOUS** and **NEXT** focus events
    - Activate the focused element using the **ACTIVATE** focus event

```
case EVT_KEY:
    switch(wParam) {
    case AVK_DOWN:
        nErr = IMEDIASVG_Focus(pMe->pIMediaSVG, MM_SVG_FOCUS_EVENT_NEXT);
        break;
    case AVK_UP:
        nErr = IMEDIASVG_Focus(pMe->pIMediaSVG, MM_SVG_FOCUS_EVENT_PREVIOUS);
        break;
    case AVK_SELECT:
        nErr = IMEDIASVG_Focus(pMe->pIMediaSVG, MM_SVG_FOCUS_EVENT_ACTIVATE);
        break;
    }
```

## Integrating SVG into BREW Application continued

- Release resources when the application exits

```
case EVT_KEY:
    switch(wParam)
    {
        case AVK_END:
            nErr = IMEDIA_Stop(pMe->pIMediaSVG) ;
            break;
    }
}
```

# Compile and Test Application

- Compile BREW application
  - No special compilation requirement to compile a BREW application using SVG functionality.
  - Only require BREW SVG header present during compilation.
    - Compile DLL file for BREW simulation
    - Compile MOD file (ARM™ compilation) for handset
- Test application on BREW Simulator
  - BREW SDK® SVG Extension must be installed to preview SVG in Simulator
- Test application on device
  - Verify that SVG is supported
  - Verify that SVG graphics are not overly stretched/compressed
  - Verify performance in both frame rate and interactivity

# Roadmap

- SVG is already shipping with most CDMA and UMTS chipsets
  - SVGT 1.1+ as of July 2005
  - SVGT 1.2 as of June 2006
- Enhancements for Q3 2008
  - Accelerated SVG on OpenVG
    - Utilizes Qualcomm's existing OpenGL ES graphics hardware
    - Increased frame rate with lower power and CPU usage
    - More concurrent capabilities and better interactivity
  - ECMA Scripting bindings support
    - SpiderMonkey and WebKit scripting engines
  - JSR 287
    - Our SVG engine is compliant with JSR 287 requirements
  - Integrated audio & video
    - OEMs can choose from all video and audio formats supported on our chipsets
  - External font engine support
    - OEMs can optionally choose to supply their own font engine for use with SVG

# For Further Information

- For SVG and CMX support
  - Email [cmx.contact@qualcomm.com](mailto:cmx.contact@qualcomm.com)
    - Ask questions
    - Get access to CMX Developer's Connection
  - CMX Developer's Connection site
    - SVG Developer Guide
    - Sample content
    - Regional handset lists
      - Indicate extent of support for SVG and CMX
- For BREW support
  - Email [brew-support@qualcomm.com](mailto:brew-support@qualcomm.com)



Thank you!

