



Animating Thousands of Graphics with ArcGIS Runtime SDK for Java

Mark Baird and Vijay Gandhi

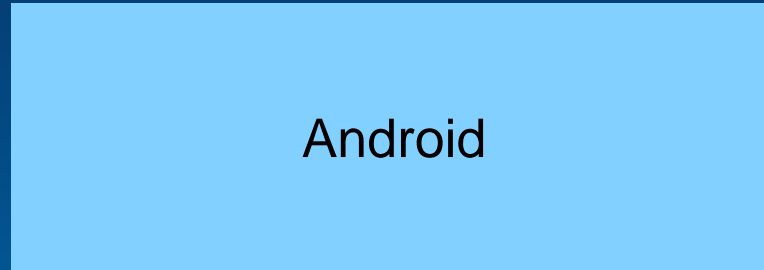
Technical Workshop

Overview

- **Runtime architecture**
- **SDK basics: graphics, features, graphics layers**
- **Demo 1: Rendering graphics**
- **Demo 2: Adding graphics**
- **Demo 3: Moving graphics**
- **Top tips!**

Runtime architecture

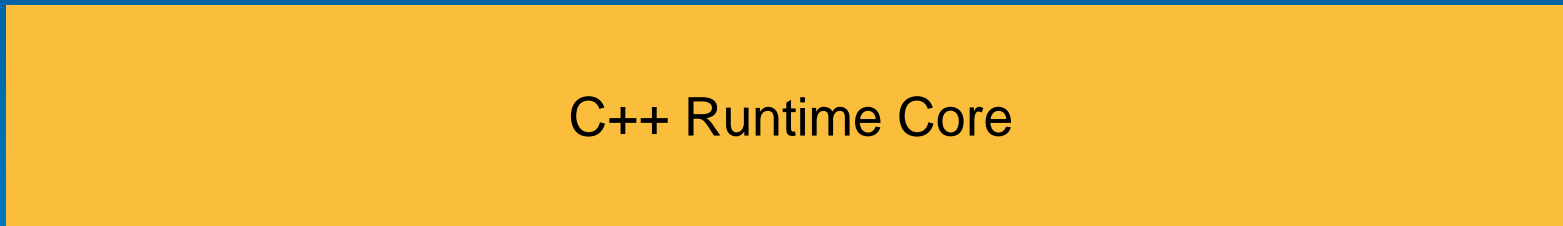
developer



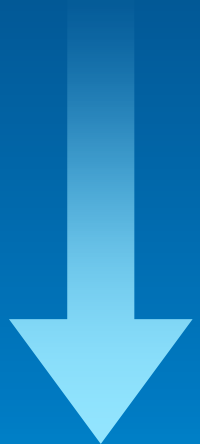
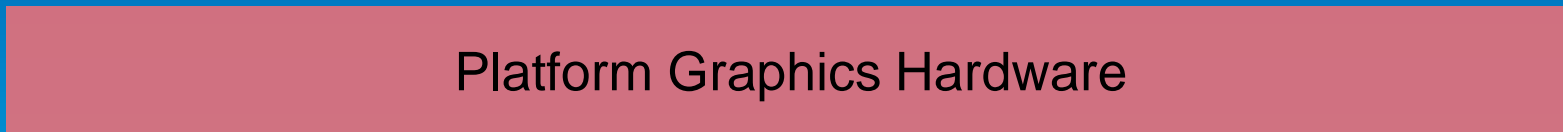
SDKs
Platform-specific



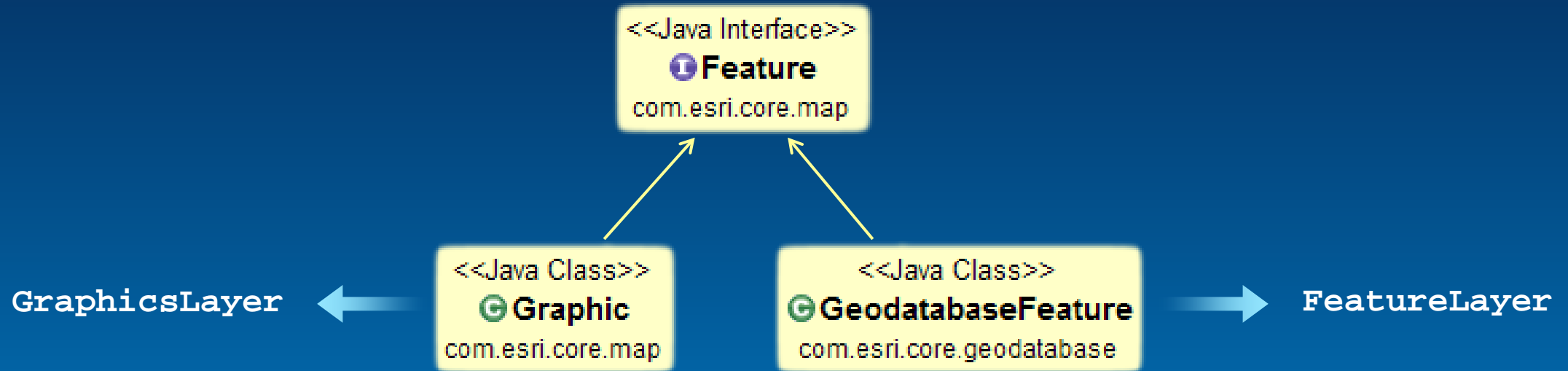
Interop layer



Native Code
Cross-platform
C++



Graphics and features



Graphic	GeodatabaseFeature
stored in memory on the client	stored in a dataset (geodatabase)
displayed in a GraphicsLayer	displayed in a FeatureLayer
can have mixed geometry type layer	one geometry type per layer



For animating moving objects, use Graphic

Graphics layer basics

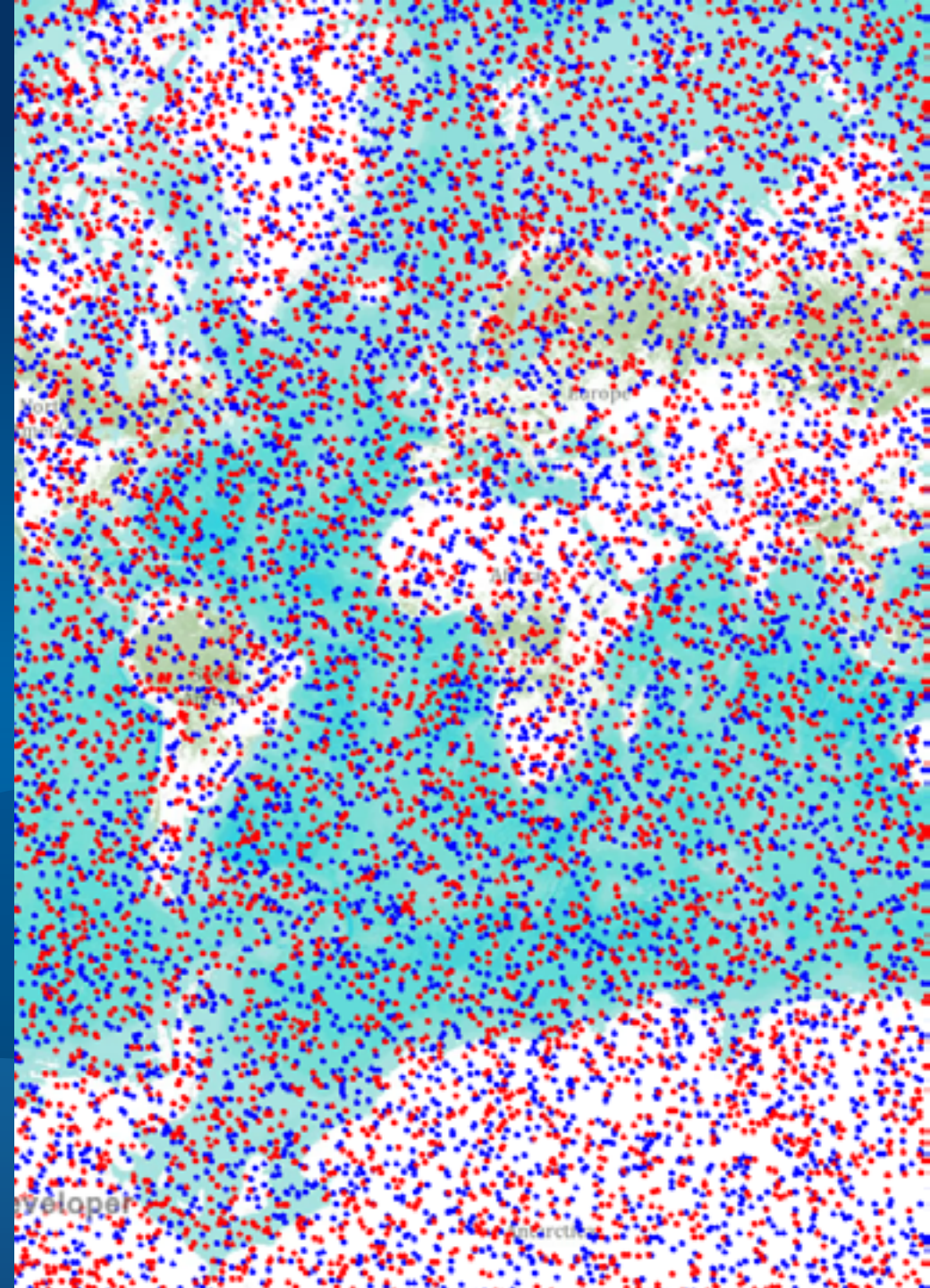
- API class is **GraphicsLayer**
- A graphics layer contains graphics (you guessed it!)
- **Static** and **Dynamic** rendering modes
- **Graphic class is immutable: so don't hold references to Graphic objects**
- **Update / move / remove graphics using methods on GraphicsLayer**
- **Work with graphics via the layer using their unique ID**

```
id = addGraphic(Graphic)
graphic = getGraphic(id)
...
updateGraphic(id, Graphic)
updateGraphic(id, Symbol)
updateGraphic(id, Geometry)
...
removeGraphic(id)
setGraphicVisible(id, visible)
...
```

DEMO

Rendering graphics

Mark Baird



Static vs Dynamic - summary

Static	Dynamic
+ Volume of graphics has little impact on frame render time (scales well)	- Volume of graphics has direct impact on (GPU) resources
- Rendering graphic updates is CPU / system memory intensive	+ Individual graphics changes can be efficiently applied directly to GPU state
Use for static graphics, complex geometries	Good in most cases, especially moving objects

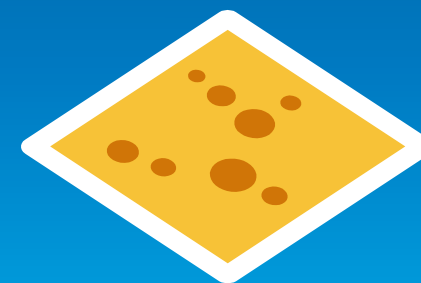


Java SE: use Dynamic mode unless you see performance issues

DEMO

Adding graphics

Vijay Gandhi



Renderer vs Symbol

Symbol (no renderer): Time to add 10k graphics: **17s**

Memory:

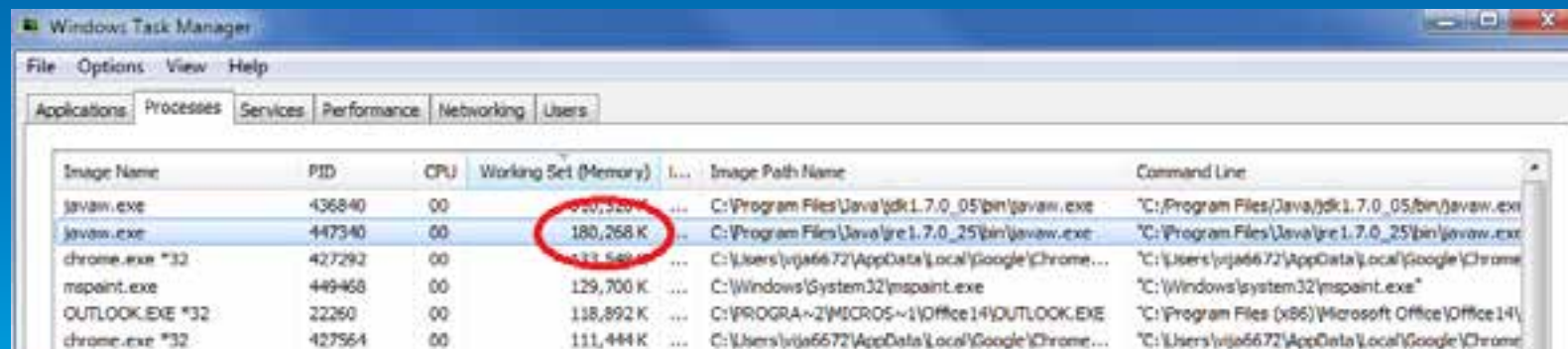


A screenshot of the Windows Task Manager 'Processes' tab. The 'Working Set (Memory)' column shows the memory usage for various processes. Two instances of 'javaw.exe' are highlighted with a red circle, showing memory usage of 358,192 K and 133,156 K.

Image Name	PID	CPU	Working Set (Memory)	Image Path Name	Command Line
javaw.exe	436840	00	358,192 K	C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe	"C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe
javaw.exe	440300	00	133,156 K	C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe	"C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe
chrome.exe *32	427292	00	118,768 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome
OUTLOOK.EXE *32	22260	00	111,440 K	C:\PROGRA~2\MICROS~1\Office14\OUTLOOK.EXE	"C:\Program Files (x86)\Microsoft Office\Office14\
chrome.exe *32	427564	00	99,576 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome
explorer.exe	6636	00	89,732 K	C:\Windows\explorer.exe	"C:\Windows\Explorer.EXE
dwm.exe	6776	00		C:\Windows\System32\dwm.exe	"C:\Windows\system32\Dwm.exe"

~360,000K

Renderer set: Time to add 10k graphics: **0.122s**



A screenshot of the Windows Task Manager 'Processes' tab. The 'Working Set (Memory)' column shows the memory usage for various processes. Two instances of 'javaw.exe' are highlighted with a red circle, showing memory usage of 180,268 K and 133,580 K.

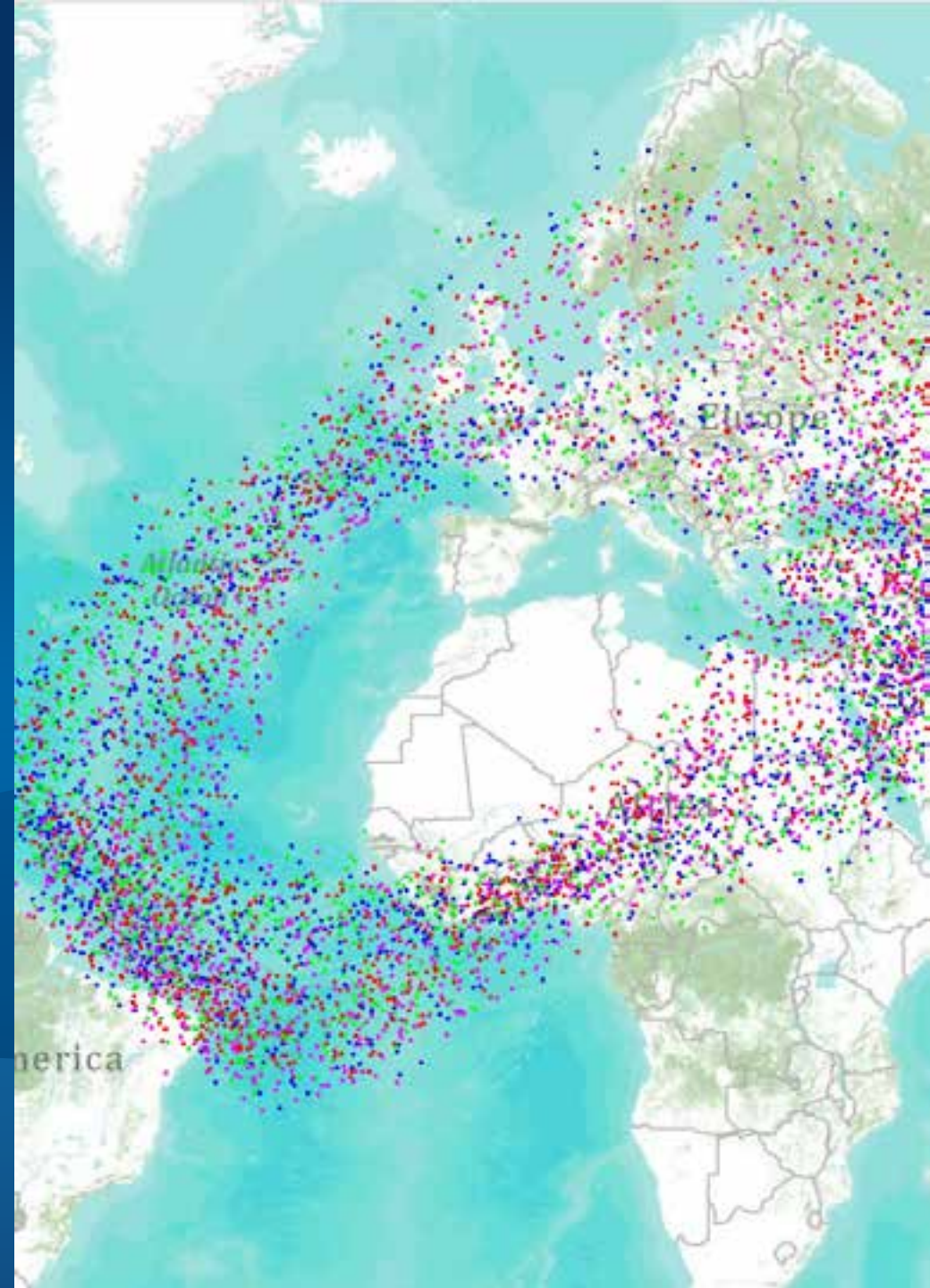
Image Name	PID	CPU	Working Set (Memory)	Image Path Name	Command Line
javaw.exe	436840	00	180,268 K	C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe	"C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe
javaw.exe	447340	00	133,580 K	C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe	"C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe
chrome.exe *32	427292	00	129,700 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome
mspaint.exe	449468	00	118,892 K	C:\Windows\System32\mspaint.exe	"C:\Windows\system32\mspaint.exe"
OUTLOOK.EXE *32	22260	00	111,444 K	C:\PROGRA~2\MICROS~1\Office14\OUTLOOK.EXE	"C:\Program Files (x86)\Microsoft Office\Office14\
chrome.exe *32	427564	00		C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome

~180,000K

DEMO

Moving graphics

Mark Baird and Vijay Gandhi



Top tips

- Move point graphics using **movePointGraphic** – optimized in Runtime Core

```
graphicsLayer.movePointGraphic(id, newPointLocation);
```

- Use a renderer on the graphics layer rather than setting individual symbols on graphics

```
graphicsLayer.setRenderer(myRenderer);
```

- Use bulk **addGraphics** method when adding many graphics at the same time

```
graphicsLayer.addGraphics(Graphics[] graphics);
```

More tips for performance

- **Split dynamic data from static data**
 - Reduce load on dynamic rendering pipeline
- **Keep different geometry types on different layers**
 - And split polygon fills from polygon outlines if possible
- **Use multiple graphics layers**
 - 5 layers of 100k point graphics will scale and perform better than 500k on one layer
- **Set scale thresholds on the layer**
 - only display the relevant subset of graphics

Questions?



ArcGIS Runtime SDK sessions Thursday

Session Name	Time	Location
ArcGIS Runtime SDKs: The Road Ahead	1:30pm – 2:45pm	Room 07 A/B

Thank you...

- **Please fill out the session survey:**

Offering ID: 1493

Online – www.esri.com/ucsessionsurveys

Paper – pick up and put in drop box



Understanding our world.