



Esri International Developer Summit
Palm Springs, CA

Animating Thousands of Graphics and Features with ArcGIS Runtime SDK for Java

Vijay Gandhi and Elise Acheson

Demo Source code:

<https://github.com/Esri/arcgis-runtime-samples-java/tree/master/dev-summit-2014>

Video Recording:

<http://video.esri.com>

Outline

- 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



Java SE

Android

SDKs
Platform-specific

JNI (Java Native Interface)

Interop layer

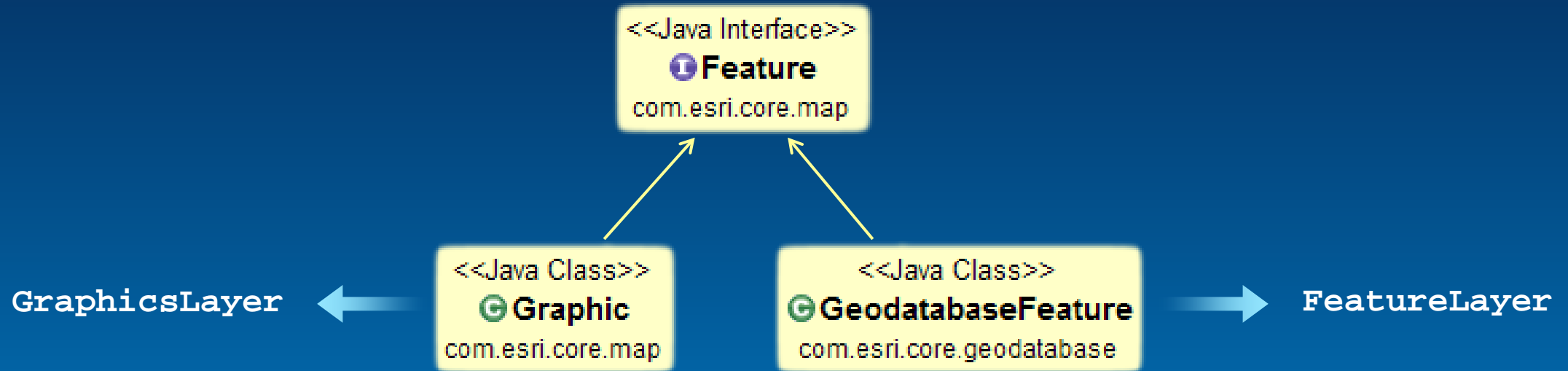
C++ Runtime Core

Native Code
Cross-platform
C++

Platform Graphics Hardware



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

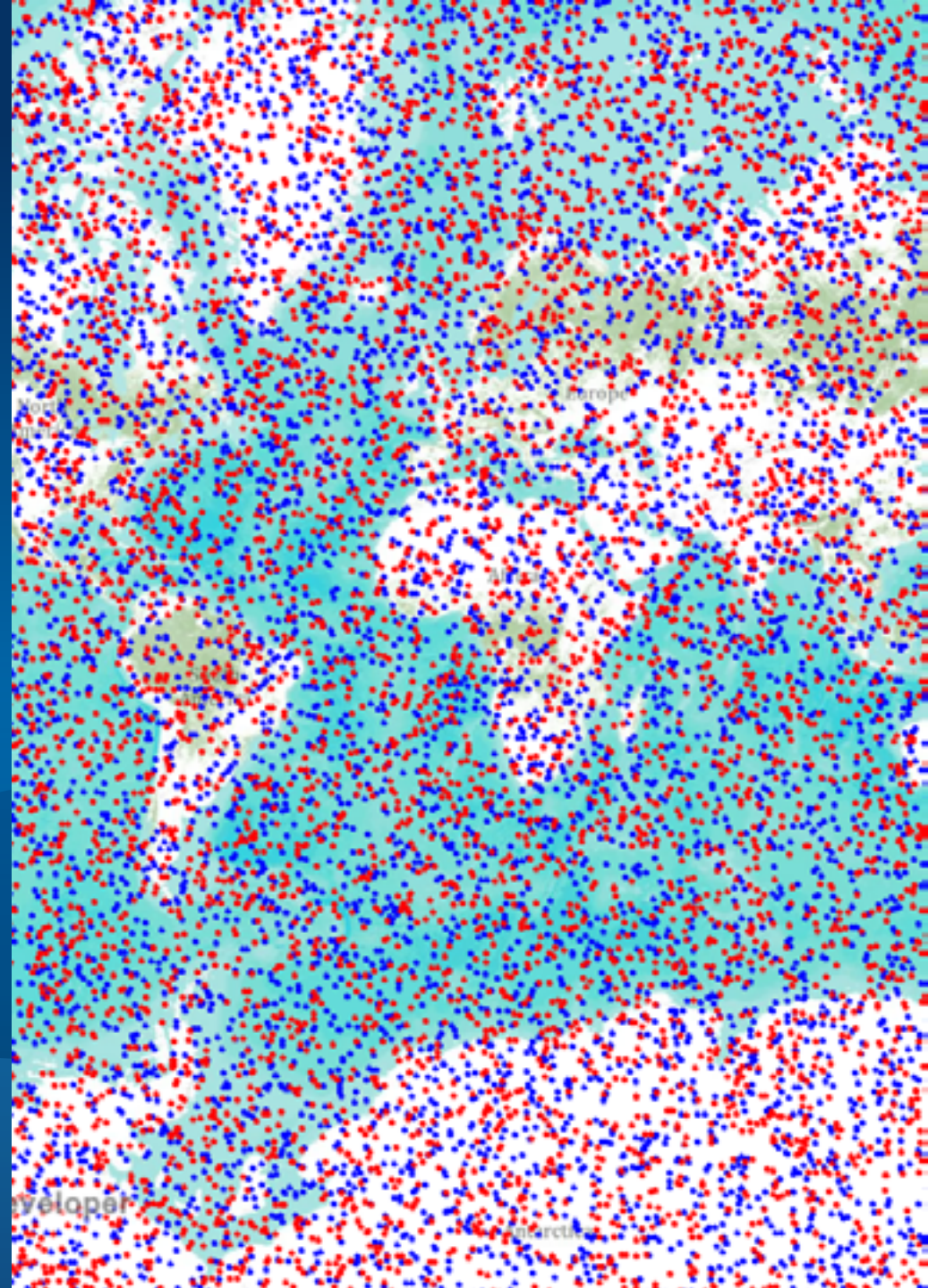
- API class is **GraphicsLayer**
- A graphics layer contains graphics (you guessed it!)
- **Static** and **Dynamic** rendering modes
- Graphic class is immutable: so don't hold references to Graphic objects
- Update / move / remove graphics using methods on GraphicsLayer
- Work with graphics via the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

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

DEMO

Rendering graphics

Elise Acheson



Static vs Dynamic - summary

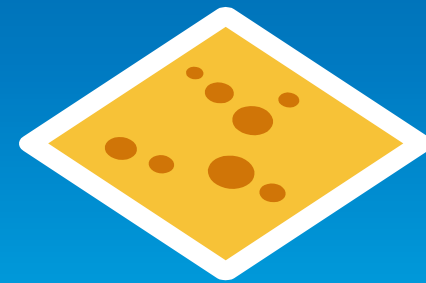
Static	Dynamic
+ Volume of graphics has little impact on frame render time (scales well)	- Volume of graphics has direct impact on (GPU) resources
- Rendering graphic updates is CPU / system memory intensive	+ Individual graphics changes can be efficiently applied directly to GPU state
Use for static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

 **Java SE: use Dynamic mode unless you see performance issues**

DEMO

Adding graphics

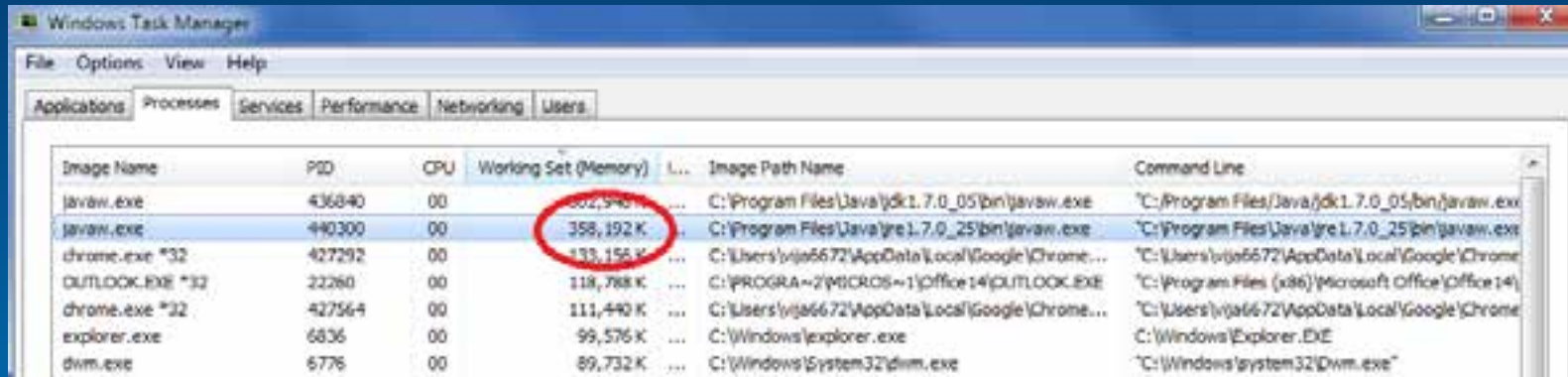
Vijay Gandhi



Renderer vs Symbol

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

Memory:

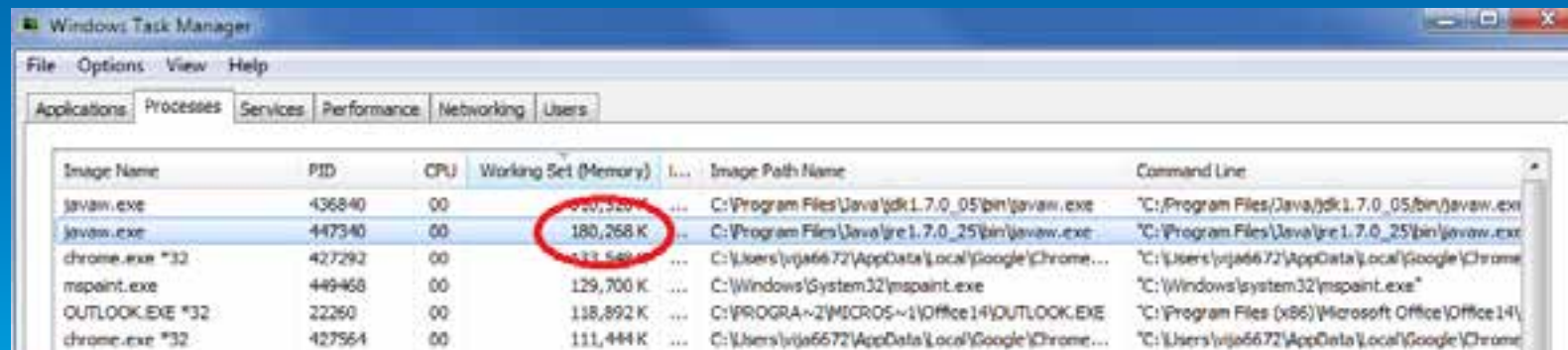


A screenshot of the Windows Task Manager 'Processes' tab. The table shows the following data:

Image Name	PID	CPU	Working Set (Memory)	Image Path Name	Command Line
javaw.exe	436840	00	302,540 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	358,192 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	133,156 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome...
OUTLOOK.EXE *32	22260	00	118,768 K	C:\PROGRA~2\MICROS~1\Office14\OUTLOOK.EXE	"C:\Program Files (x86)\Microsoft Office\Office14\...
chrome.exe *32	427564	00	111,440 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome...
explorer.exe	6636	00	99,576 K	C:\Windows\explorer.exe	"C:\Windows\Explorer.EXE"
dwm.exe	6776	00	89,732 K	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 table shows the following data:

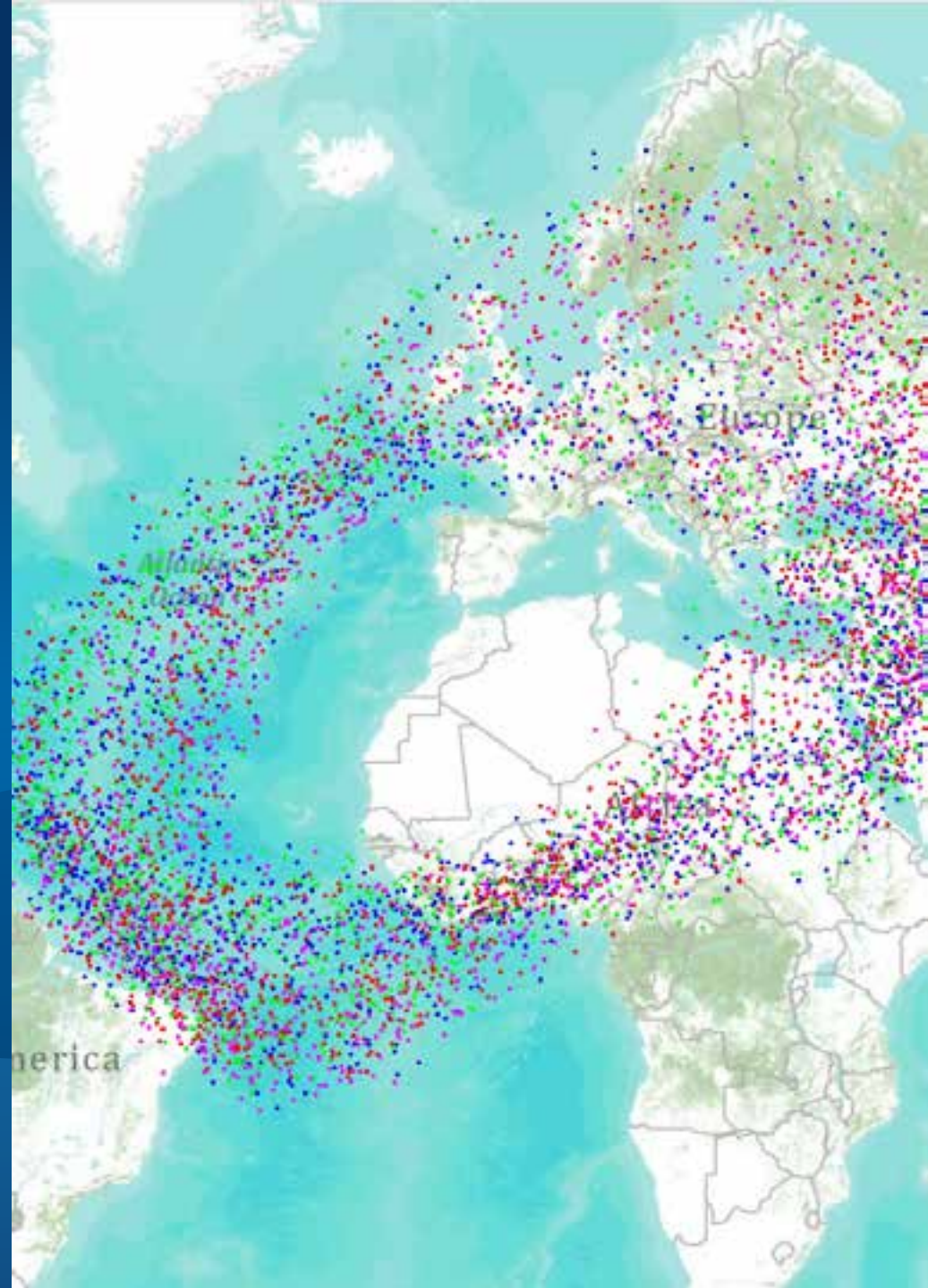
Image Name	PID	CPU	Working Set (Memory)	Image Path Name	Command Line
javaw.exe	436840	00	300,520 K	C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe	"C:\Program Files\Java\jdk1.7.0_05\bin\javaw.exe"
javaw.exe	447340	00	180,268 K	C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe	"C:\Program Files\Java\jre1.7.0_25\bin\javaw.exe"
chrome.exe *32	427292	00	133,580 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome...
mspaint.exe	449468	00	129,700 K	C:\Windows\System32\mspaint.exe	"C:\Windows\system32\mspaint.exe"
OUTLOOK.EXE *32	22260	00	118,892 K	C:\PROGRA~2\MICROS~1\Office14\OUTLOOK.EXE	"C:\Program Files (x86)\Microsoft Office\Office14\...
chrome.exe *32	427564	00	111,444 K	C:\Users\vsja6672\AppData\Local\Google\Chrome...	"C:\Users\vsja6672\AppData\Local\Google\Chrome...

~180,000K

DEMO

Moving graphics

Elise Acheson / 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(Graphic[] 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?

Next sessions:

- Wednesday, 4:00pm - 5:00pm, Smoketree F

Building Java Apps using ArcGIS Runtime SDK

- Thursday 8:30am - 9:30am, Primrose A

Road Ahead for ArcGIS Runtime SDKs

- Thursday 10:00am – 11:00am, Primrose A

Everything (or Anything) You Wanted to Know about ArcGIS Runtime SDKs



Understanding our world.