Make your iOS Apps come alive with Map Animations

Nicholas Furness
Agenda

• Point animation

• Polyline animation

• Viewpoint animation
Point Animation
Animate along a path

Demo
Animate along path

Frame 1
Frame 100
Frame 200
Frame 300
Frame 400
Animate along path

**AnimateAlongPathHelper Class**
- polyline: AGSPolyline
- animatingGraphic: AGSGraphic
- speed: Double
+ startAnimation()
+ stopAnimation()

**AnimateAlongPathHelperDelegate**
- animateAlongPathHelperDidFinish()
// create an instance of helper class
// use route geometry as the polyline, car graphic as the animating graphic
animateAlongPathHelper = AnimateAlongPathHelper(polyline: routeGeometry, 
                                                animatingGraphic: carGraphic, speed: 2000)

// start the animation
animateAlongPathHelper.startAnimation()
```swift
func startAnimation() {
    let length = AGSGeometryEngine.length(of: polyline)
    let duration = length/speed
    let startTime = Date()

    AnimationManager.animate(animationBlock: { [weak self] () -> Bool in
        guard let polyline = self?.polyline else {
            return true
        }
    }, completion: { [weak self] in
        if let me = self {
            self?.delegate?.animateAlongPathHelperDidFinish?(me)
        }
    })

    let doneFactor = Date().timeIntervalSince(startTime) / duration
    let newLocation = AGSGeometryEngine.point(along: polyline,
                                             distance: length * doneFactor)

    self?.animatingGraphic.geometry = newLocation

    return doneFactor >= 1
}
```
func startAnimation() {
    let length = AGSGeometryEngine.length(of: polyline)
    let duration = length/speed
    let startTime = Date()

    AnimationManager.animate(animationBlock: { [weak self] () -> Bool in
        guard let polyline = self?.polyline else {
            return true
        }

        let doneFactor = Date().timeIntervalSince(startTime) / duration
        let newLocation = AGSGeometryEngine.point(along: polyline,
            distance: length * doneFactor)

        self?.animatingGraphic.geometry = newLocation

        return doneFactor >= 1
    }, completion: { [weak self] in
        if let me = self {
            self?.delegate?.animateAlongPathHelperDidFinish?(me)
        }
    })
}
```swift
func startAnimation() {
    let length = AGSGeometryEngine.length(of: polyline)
    let duration = length/speed
    let startTime = Date()

    AnimationManager.animate(animationBlock: { [weak self] () -> Bool in
        guard let polyline = self?.polyline else {
            return true
        }

        let doneFactor = Date().timeIntervalSince(startTime) / duration
        let newLocation = AGSGeometryEngine.point(along: polyline,
            distance: length * doneFactor)

        self?.animatingGraphic.geometry = newLocation

        return doneFactor >= 1
    }, completion: { [weak self] in
        if let me = self {
            self?.delegate?.animateAlongPathHelperDidFinish?(me)
        }
    })
}
```
Animation Manager

AnimationManager

+ animate(<animation closure>)
+ pause()
+ resume()

- Singleton
- Internal Timer
- Array of closures
  - Each closure updates an animated graphic
- 60 times a second
Draw polyline animation

Demo
Polyline animation
Remember animating along the path?

Frame 1

Frame 100

Frame 200

Frame 300

Frame 400
Trace polyline animation

Frame 1
Frame 100
Frame 200
Frame 300
Frame 400
Draw polyline animation

**AnimateLineTraceHelper Class**

- polyline: AGSPolyline
- animatingGraphic: AGSGraphic
- speed: Double

+ startAnimation()
+ stopAnimation()
func startAnimation() {
    let length = AGSGeometryEngine.length(of: polyline)
    let duration = length/speed
    let startTime = Date()

    AnimationManager.animate(animationBlock: { () -> Bool in
        let doneFactor = Date().timeIntervalSince(startTime) / duration
        guard let newLocation = AGSGeometryEngine.point(along: self.polyline, distance: length * doneFactor) else {
            print("New location is not a valid point! Done factor = \$(doneFactor)")
            return true
        }
        self.polylineBuilder.add(newLocation)

        // assign the geometry to the polyline graphic
        self.animatingGraphic.geometry = self.polylineBuilder.toGeometry()
    }, completion: nil)
}
```swift
func startAnimation() {
    let length = AGSGeometryEngine.length(of: polyline)
    let duration = length/speed
    let startTime = Date()

    AnimationManager.animate(animationBlock: { () -> Bool in

        let doneFactor = Date().timeIntervalSince(startTime) / duration
        guard let newLocation = AGSGeometryEngine.point(along: self.polyline, distance: length * doneFactor) else {
            print("New location is not a valid point! Done factor = \(doneFactor)")
            return true
        }

        self.polylineBuilder.add(newLocation)

        // assign the geometry to the polyline graphic
        self.animatingGraphic.geometry = self.polylineBuilder.toGeometry()

        return doneFactor >= 1
    }, completion: nil)
}
```
Applications

Demo
Viewpoint Animation
Viewpoint Animation Demo
Viewpoint animation

AGSMapView
- `setViewpoint(_: duration: completion:)`
- `setViewpoint(_: duration: curve: completion:)`

AGSViewpoint
- `initWithCenter(_: scale:)`
- `initWithCenter(_: scale: rotation:)`
- `initWithTargetExtent(_:)`
- `initWithTargetExtent(_: rotation:)`
- `initWithLatitude(_: longitude: scale:)`
Viewpoint animation

AGSMapView

- setViewpointCenter(_: completion:)
- setViewpointCenter(_: scale: completion:)
- setViewpointScale(_: completion:)
- setViewpointRotation(_: completion:)
- setViewpointGeometry(_: completion:)
- setViewpointGeometry(_: padding: completion:)
Turn by turn directions

Demo
Summary

- Moving a point along a line
- Tracing a line
- Animating the map’s viewpoint
Tips

• Performance
  - Consider AGSGraphicsOverlay rendering mode .dynamic
  - Use a renderer (instead of setting a symbol on each graphic)
  - Be wary of too many Timers (AnimationManager keeps just one)
  - Consider if you should Generalize complex geometries
  - The AGSGeometryEngine is your friend

• UX
  - Show direction to add context
  - Don’t go 🍌 🍌 🍌. Too much animation can be a distraction
One more thing…
# Scene Camera animation

**Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(unavailable(&quot;init is not available.&quot;))</td>
<td><strong>attribute</strong></td>
</tr>
<tr>
<td>(instancetype)</td>
<td>initWithTargetGeoElement:distance:</td>
</tr>
<tr>
<td>(id&lt; AGSCancelable &gt;)</td>
<td>moveCameraWithDistanceDelta:headingDelta:pitchDelta:duration:completion:</td>
</tr>
<tr>
<td>(id&lt; AGSCancelable &gt;)</td>
<td>setTargetOffsetX:targetOffsetY:targetOffsetZ:duration:completion:</td>
</tr>
</tbody>
</table>

**Class Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(instancetype)</td>
<td>+ orbitGeoElementCameraControllerWithTargetGeoElement:distance:</td>
</tr>
</tbody>
</table>
Please Take Our Survey!

Download the Esri Events app and find your event

Select the session you attended

Scroll down to the “Feedback” section

Complete Answers, add a Comment, and Select “Submit”
Demo project is posted on GitHub:

github.com/nixta/iOS-runtime-animations