Building 3D Apps with ArcGIS API for JavaScript

Veronika Landers - Yannik Messerli - Johannes Schmid
3D GIS across industries
Agenda

- Introduction
- Your first 3D app
- Working with the API
- Demos
Introduction

Veronika Landers
The ArcGIS Platform

ArcGIS Online
ArcGIS Enterprise

Server
Online Content and Services

Desktop  Device  Web
The ArcGIS 3D Platform
Out-of-the-box 3D Web Apps

Scene Viewer

Story Maps

Web AppBuilder
Building Apps with the ArcGIS 3D Platform

ArcGIS Pro
CityEngine
ArcGIS Online
ArcGIS Enterprise

Scene Viewer
Story Maps
Web AppBuilder

ArcGIS API for JavaScript

Desktop
Device
Web

Server
Online Content and Services
Custom 3D Web Apps

Add news/social feeds,

filter/reporting functionality,

custom UI elements, etc …
ArcGIS API for JavaScript
ArcGIS API for JavaScript

- Visual mapping (2D & 3D), components and widgets
- Support for various different layer types (data sources)
- Integration with the ArcGIS platform
  (security, sign-in, premium services, ...)

Get it today

Find doc & samples to get started at https://developers.arcgis.com/javascript
Desktop **3D** Requirements

- Modern hardware, especially Graphics Card
- Latest web browsers with WebGL support
  - Chrome
  - Edge / Internet Explorer 11* (*WebGL not optimized)
  - Firefox
  - Safari
Mobile 3D Requirements

- Mobile support introduced this year!
- Officially supported for the following devices:
  - iOS — iPhone 8, iPad Pro (Safari)
  - Android — Samsung S8/S9 (Chrome)
  - Android - Samsung Tab S3 (Chrome)
- Other devices only at your own risk :)
Content for your apps
Publish a layer in ArcGIS Pro
Publish a web scene in ArcGIS Pro
Coordinate System Support for Scenes

- Each scene has its coordinate system defined.
- Cached data needs to be provided in the defined system.
- Feature and dynamic data is projected on the fly.
- Coordinate systems define the type of scenes and its capabilities.
Global scenes

Visualize data on a full globe
Global scenes

Choose one of two geographic coordinate systems

- WebMercator (wkid: 3857) - ideal to use Esri provided cached data
- WGS84 (wkid: 4326)
Local scenes

Visualize data in a local planar way
Local scenes
Choose one of the following options

- WebMercator (wkid: 3857)
- any projected coordinate system

Clip to your area of interest
Web Scenes Concept

- Vehicle for cross platform 3D capabilities
- Collection of layers, environment settings, slides
- Read and write Web Scenes across the ArcGIS platform
Your first 3D App

with the ArcGIS API for JavaScript

Yannik Messerli
Steps
1. Loading the webscene
2. Add a layer
3. Add a widget
4. Configure popups
Creating basic HTML file

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="/js.arcgis.com/4.8/esri/css/main.css">
  <script src="/js.arcgis.com/4.8/"></script>
  <title>Create a 3D map</title>
</head>
<body>
  <script>
    require(
      [
        "esri/WebScene",
        "esri/views/SceneView",
        "dojo/domReady!"
      ], function(WebScene, SceneView) {
        var scene = new WebScene({
          portalItem: {
            id: "e5dab9bfc78a46d08276b185026fc20a"
          }
        });
        var view = new SceneView({
          container: "viewDiv",
          map: scene
        });
      });
  </script>
</body>
</html>
```
Reference the ArcGIS API for JavaScript

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="/js.arcgis.com/4.8/esri/css/main.css">
  <script src="/js.arcgis.com/4.8/"></script>
  <title>Create a 3D map</title>
</head>
<body>
<script>
  require([
    "esri/WebScene",
    "esri/views/SceneView",
    "dojo/domReady!"
  ], function(WebScene, SceneView) {
    var scene = new WebScene({
      portalItem: {
        id: "e5dab9bfc78a46d08276b185026fc20a"
      }
    });
    var view = new SceneView({
      container: "viewDiv",
      map: scene
    });
  });
</script>
```
Load the module

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="/js.arcgis.com/4.8/esri/css/main.css">
  <script src="/js.arcgis.com/4.8/"></script>
  <title>Create a 3D map</title>
</head>
<body>
  <script>
    require(['
      "esri/WebScene",
      "esri/views/SceneView",
      "dojo/domReady!"
    ], function(WebScene, SceneView) {
      var scene = new WebScene({
        portalItem: {
          id: "e5dab9bfc78a46d08276b185026fc20a"  
        }
      });
      var view = new SceneView({
        container: "viewDiv",
        map: scene
      });
    });
  </script>
</body>
</html>
```
Instantiate the webscene

```javascript
var view = new SceneView({
  container: "viewDiv",
  map: scene
});
```
Create the view

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="//js.arcgis.com/4.8/esri/css/main.css">
  <script src="//js.arcgis.com/4.8/"></script>
  <title>Create a 3D map</title>
</head>
<body>
<script>
  require([  
    "esri/WebScene",
    "esri/views/SceneView",
    "dojo/domReady!"
  ], function(WebScene, SceneView) {
    var scene = new WebScene({
      portalItem: {
        id: "e5dab9bfc78a46d08276b185026fc20a"
      }
    });

    var view = new SceneView({
      container: "viewDiv",
      map: scene
    });
  });
</script>
```
Your first 3D app!

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="/js.arcgis.com/4.8/esri/css/main.css">
<script src="/js.arcgis.com/4.8/"/>
<title>Create a 3D map</title>
</head>
<body>
<script>
require(["esri/WebScene", "esri/views/SceneView", "dojo/domReady!"]
, function(WebScene, SceneView) {
var scene = new WebScene({
  portalItem: {
    id: "e5dab9bfc78a46d08276b185026fc20a"
  }
});
var view = new SceneView({
  container: "viewDiv",
  map: scene
});
});
</script>
<div id="viewDiv"></div>
</body>
</html>
```
Details about the **Webscene** class

- **Layers**, **Presentation**, ...
- **Basemap** is exactly the same concept as in 2D
- **Ground** defines the ground surface of the scene
## Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>CSVLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>StreamLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>MapImageLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>ImageryLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WMSLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>OpenStreetMapLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>TileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WebTileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WMTSLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>VectorTileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>ElevationLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>SceneLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>IntegratedMeshLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>PointCloudLayer</td>
<td>3D only</td>
</tr>
</tbody>
</table>
Layers

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>StreamLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>MapImageLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>ImageryLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WMSLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>OpenStreetMapLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>TileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WebTileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>WMTSLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>VectorTileLayer</td>
<td>2D &amp; 3D</td>
</tr>
<tr>
<td>ElevationLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>SceneLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>IntegratedMeshLayer</td>
<td>3D only</td>
</tr>
<tr>
<td>PointCloudLayer</td>
<td>3D only</td>
</tr>
</tbody>
</table>
Adding a layer

```javascript
require([
    "esri/WebScene",
    "esri/views/SceneView",
    "esri/layers/FeatureLayer",
    "dojo/domReady!"
], function(WebScene, SceneView, FeatureLayer) {

    var scene = new WebScene({
        portalItem: {
            id: "e5dab9bfc78a46d08276b185026fc20a"
        }
    });

    var view = new SceneView({
        container: "viewDiv",
        map: scene
    });

    var layer = new FeatureLayer({
        portalItem: {
            id: "a38c0bd41aad41d89ab2a31050ff07b1"
        }
    });

    scene.add(layer);
});
```
Adding a layer

```
require([
  "esri/WebScene",
  "esri/views/SceneView",
  "esri/layers/FeatureLayer",
  "dojo/domReady!"
], function(WebScene, SceneView, FeatureLayer) {
  var scene = new WebScene({
    portalItem: {
      id: "e5dab9bfc78a46d08276b185026fc20a"
    }
  });
  var view = new SceneView({
    container: "viewDiv",
    map: scene
  });
  var layer = new FeatureLayer({
    portalItem: {
      id: "a38c0bd41aad41d89ab2a31050ff07b1"
    }
  });
  scene.add(layer);
});
```
Widgets

Widgets are UI components that add functionalities to your scene. The API provides ready-to-use widgets, for example:

- **Legend**
- **LayerList**
- **Search**
- ...

Adding a widget

```javascript
require([  "esri/WebScene",  "esri/views/SceneView",  "esri/layers/FeatureLayer",  "esri/widgets/Search",  "dojo/domReady!" ], function(  WebScene,  SceneView,  FeatureLayer,  Search  ) {
  // ...

  var searchWidget = new Search({  
    view: view
  });
  view.ui.add(searchWidget, {  
    position: "top-right"
  });
});
```
require([  "esri/WebScene",  "esri/views/SceneView",  "esri/layers/FeatureLayer",  "esri/widgets/Search",  "dojo/domReady!" ], function(  WebScene,  SceneView,  FeatureLayer,  Search  ) {  

    // ...

    var searchWidget = new Search({  
        view: view  
    });  
    view.ui.add(searchWidget, {  
        position: "top-right"  
    });  
});
Popups

Enable on a layer:

```javascript
layer.popupEnabled = true;
```

Open it programmatically:

```javascript
view.on("click", function(event) {
  event.stopPropagation();
  view.popup.open({
    title: "Reverse geocode: 
      + event.mapPoint.longitude
      + ", 
      + event.mapPoint.latitude
      + "\}",
    location: event.mapPoint
  });
});
```
Popup templates

Display attribute using the

```javascript
var template = {
    title: "Building <b>{NAME}</b>",
    content: "This building has an energy consumption<br>
    + "of <b>{ElectricUse}</b> kBTU, for a score "
    + "of <b>{StarScore}</b>."
};
layer.popupTemplate = template;
```
Working with the API

Johannes Schmid
Architecture

Model
- Content description
- Persistence

View
- Renders the scene
- User interaction
class SceneView {
   // Camera specifies the view
   camera: Camera;

   // Animations, framing
   goTo(...);

   // Finding graphics at screen locations
   hitTest(...);

   // Converting coordinate systems
   toScreen(mapPoint: Point): ScreenPoint;
   toMap(screenPoint: ScreenPoint): Point;
}
Camera

- Primary specification of the view is the `Camera`

```javascript
class Camera {
  // The position of the camera eye in 3D space (x, y + z elevation)
  position: Point;

  // The heading angle (towards north in degrees, [0, 360]°)
  heading: number;

  // The tilt angle ([0, 180]°, with 0° straight down, 90° horizontal)
  tilt: number;
}
```
const camera = view.camera.clone();

// Increment the heading of the camera by 5 degrees
camera.heading += 5;

// Set the modified camera on the view
view.camera = camera;
```javascript
// target heading = current heading + 30
var newHeading = view.camera.heading + 30;

// go to heading preserves view.center
view.goTo({
    heading: newHeading
});

// coordinates (lon, lat) of Mount Fuji
var newCenter = [138.729050, 35.360638];
view.goTo({
    center: newCenter,
    zoom: 13
});
```
// Every time the user clicks on the map...
view.on("click", function(event) {

    // convert the screen position to map coordinates
    var position = view.toMap(event.x, event.y);

    // add a cone symbol at that location
    view.graphics.add(new Graphic({
        geometry: position,
        symbol: coneSymbol
    });
});
Working with layers

class SceneLayer {
    // Renderer assigns each feature a color and style
    renderer: Renderer;

    // Filtering
    definitionExpression: string;

    // Querying
    queryFeatures(params: Query): FeatureSet;
    queryExtent(params: Query): Extent;
    ...
}
Filtering

// only show buildings constructed before 1900
sceneLayer.definitionExpression = "CNSTRCT_YR < 1900 AND CNSTRCT_YR > 0";

// reset filter
sceneLayer.definitionExpression = null;

// only show tall buildings
sceneLayer.definitionExpression = "HEIGHTROOF > 300";
Assigning a renderer

```javascript
// draw buildings in transparent green
sceneLayer.renderer = {
  type: "simple",
  symbol: {
    type: "mesh-3d",
    symbolLayers: [{
      type: "fill",
      material: {
        color: [144, 238, 144, 0.3]
      }
    }]
  }
};

// color buildings by construction year
sceneLayer.renderer = {
  type: "simple",
  visualVariables: [{
    type: "color",
    field: "CNSTRCT_YR",
    stops: [{
      value: 1867,
      color: [69, 83, 122]
    },
    ...]
  }]
};
```
ArcGIS API for JavaScript: 3D Visualization

Thursday, 10:00 am - 11:00 am
Room 33C
There is so much more...

https://developers.arcgis.com/javascript/
ArcGIS API for JavaScript: An Introduction
Tuesday, 4:00 pm - 5:00 pm
Room 30E

ArcGIS API for JavaScript: Best Practices for Building Apps
Wednesday, 2:30 pm - 3:30 pm
Room 31A
Demos

of 3D applications

Veronika Landers - Johannes Schmid - Yannik Messerli
Related sessions

ArcGIS API for JavaScript: An Introduction
Tuesday, 4:00 pm - 5:00 pm
Room 30E

ArcGIS API for JavaScript: Best Practices for Building Apps
Wednesday, 2:30 pm - 3:30 pm
Room 31A

ArcGIS API for JavaScript: 3D Visualization
Thursday, 10:00 am - 11:00 am
Room 33C
Please Take Our Survey on the App

1. Download the Esri Events app and find your event
2. Select the session you attended
3. Scroll down to find the feedback section
4. Complete answers and select “Submit”
Topics

- Skyscraper explorer
- Hiking App
- C-Through
- Satellite map
- Landsat viewer
Skyscraper explorer
Hiking App
C-Through
Satellite map
Landsat viewer