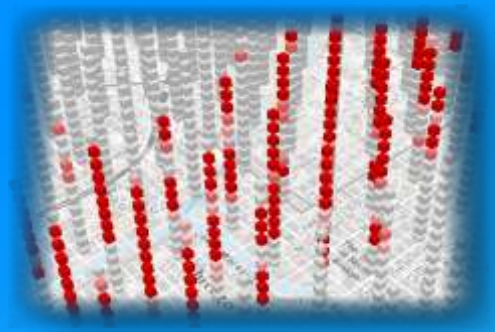
Getting Started with 3D in the ArcGIS API for JavaScript

Javier Gutierrez

Jesse van den Kieboom

2018 **Esri DEV**Summit **Conference** | Palm Springs, CA

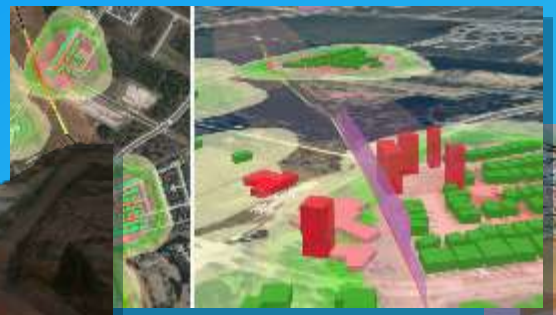
3D GIS across industries



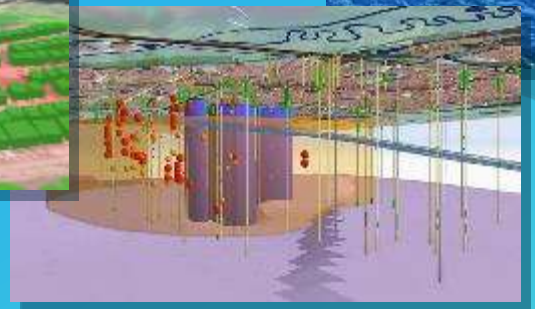
Scientific Visualization

Mining

Developing Energy resources



Transportation

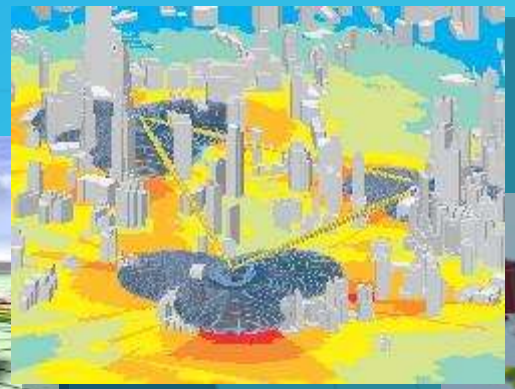


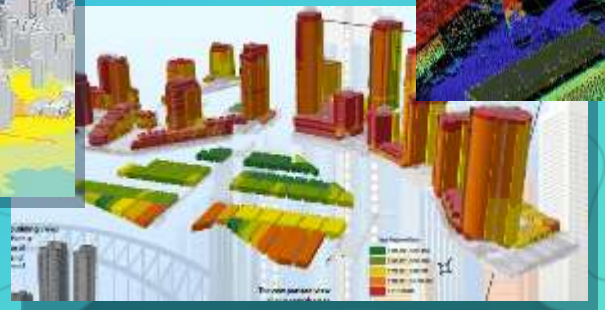
Environmental assessment

City monitoring and planning

Facilities Management

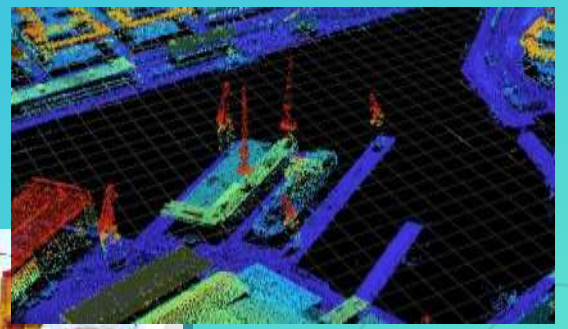
Utilities and Telecommunications





Land Management

Infrastructure



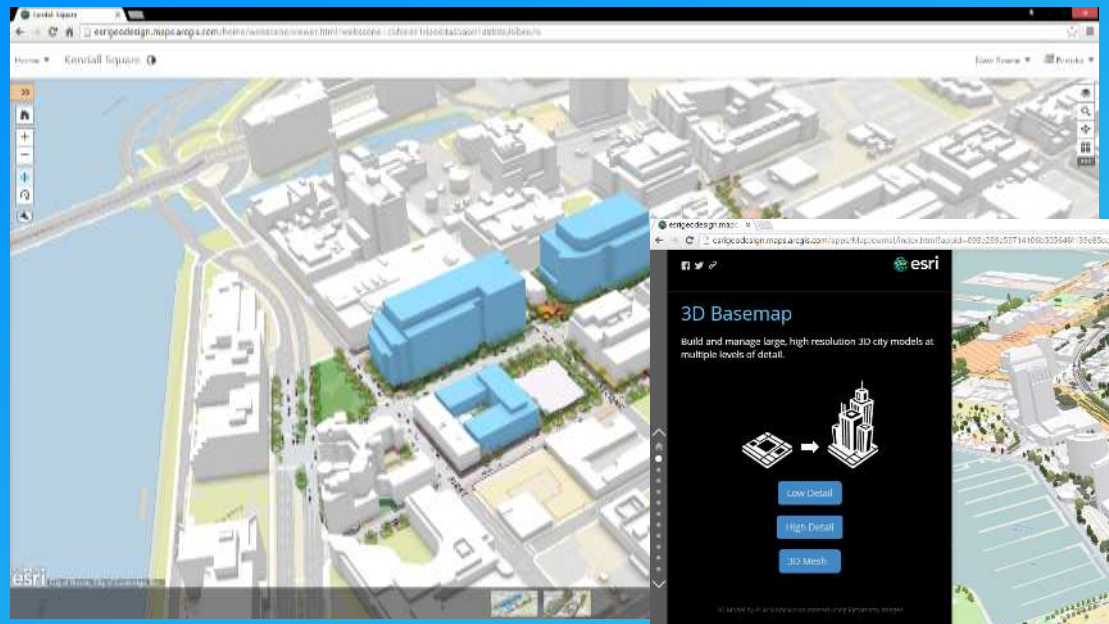
Introduction

3D in the Browser with the ArcGIS Platform

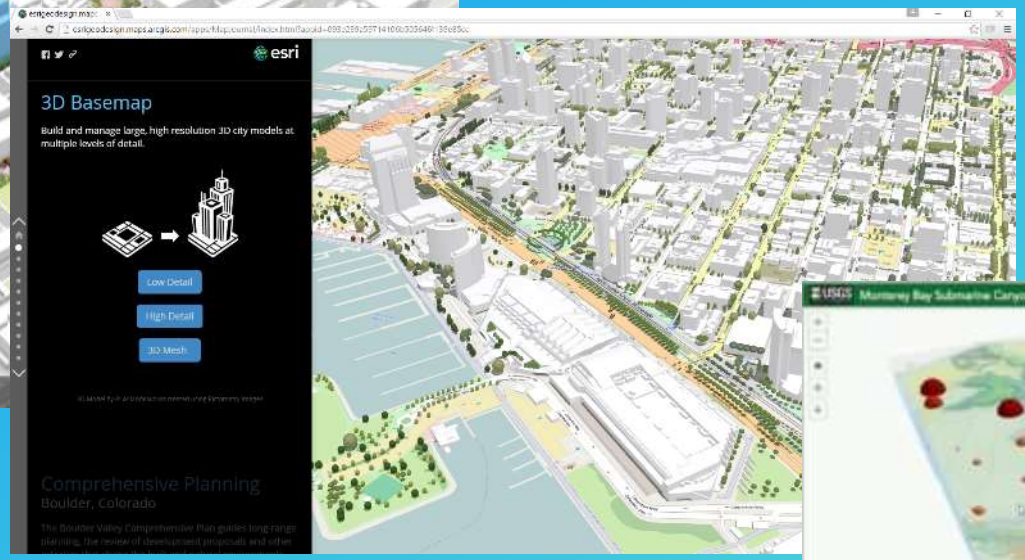
The ArcGIS 3D Platform

A System for Managing and Applying Geographic Information

Esri Provides out-of-the-box 3D Web Apps



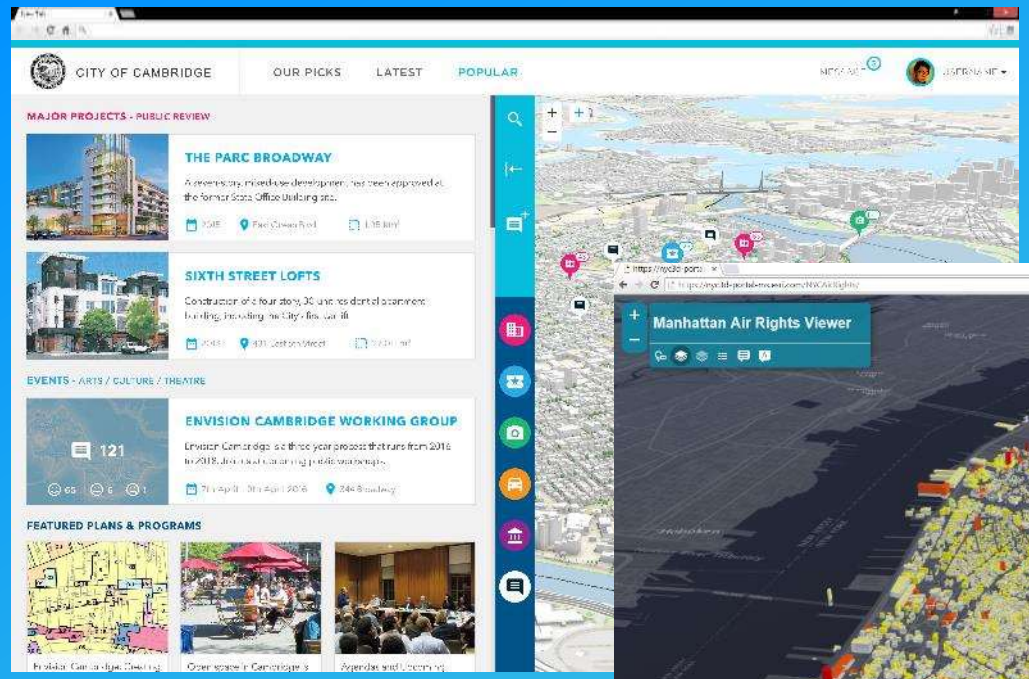
Scene Viewer



Story Maps

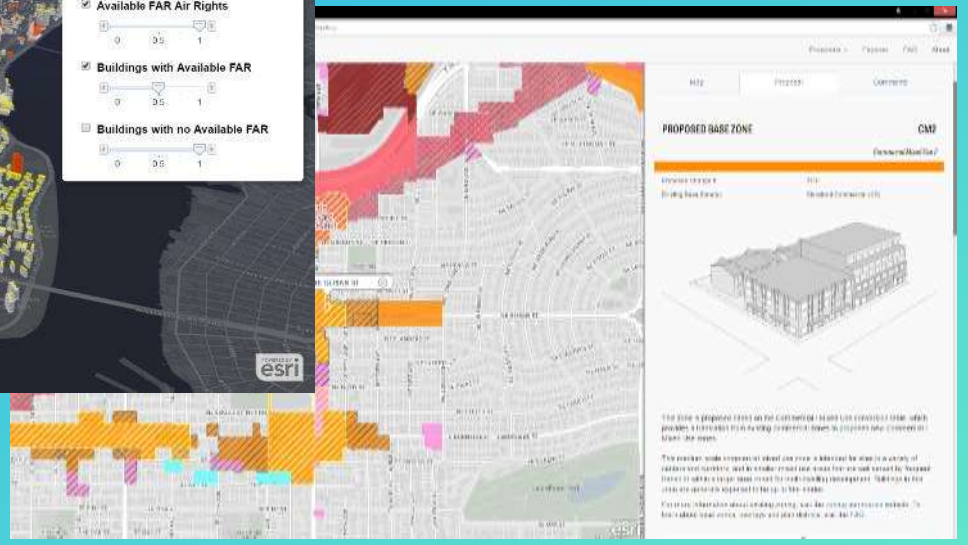
Web AppBuilder …

Build Your Own custom 3D Web Apps



Add news/social feeds,

filter/reporting functionality,

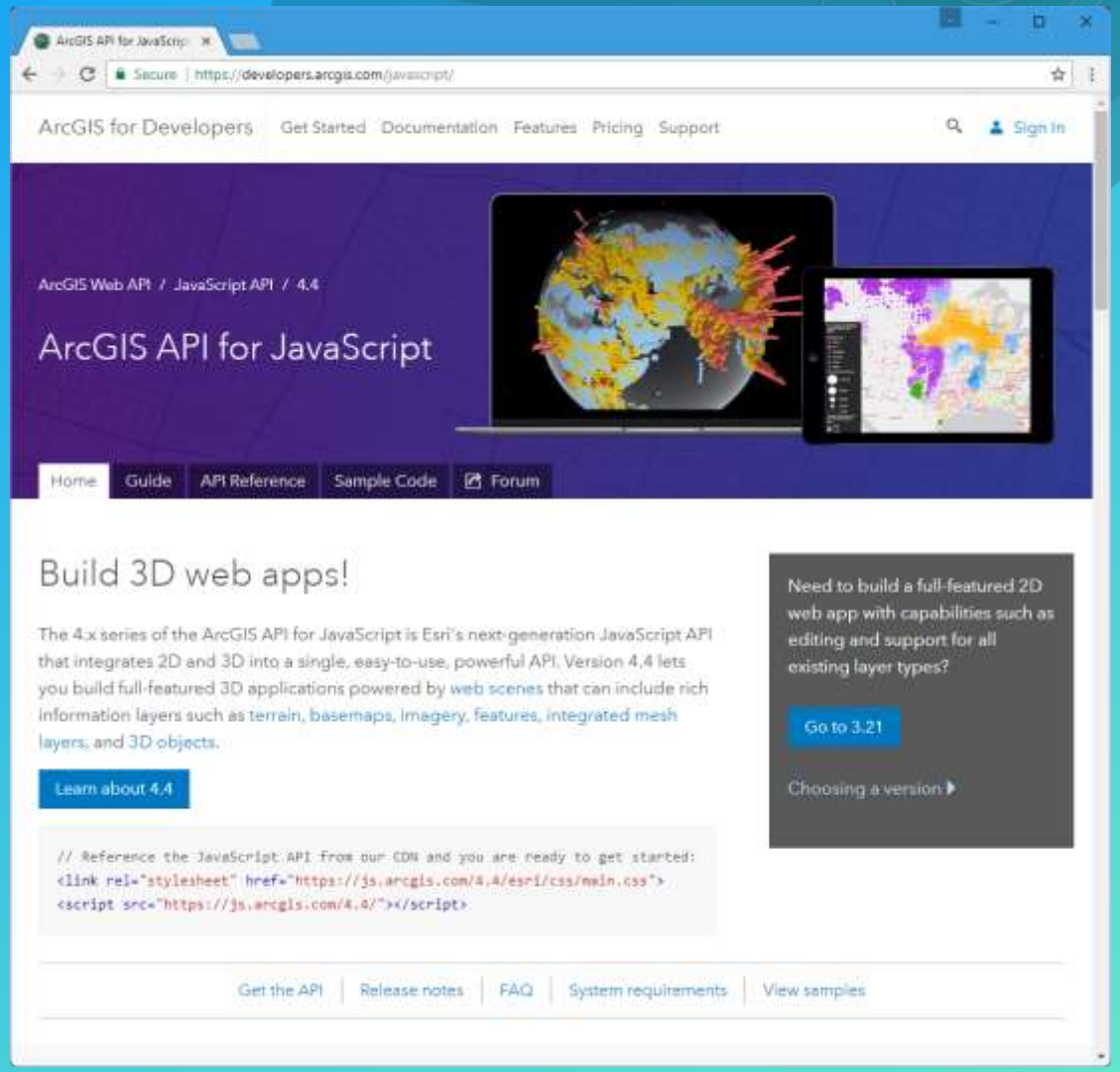


custom UI elements, etc …

The ArcGIS API for JavaScript

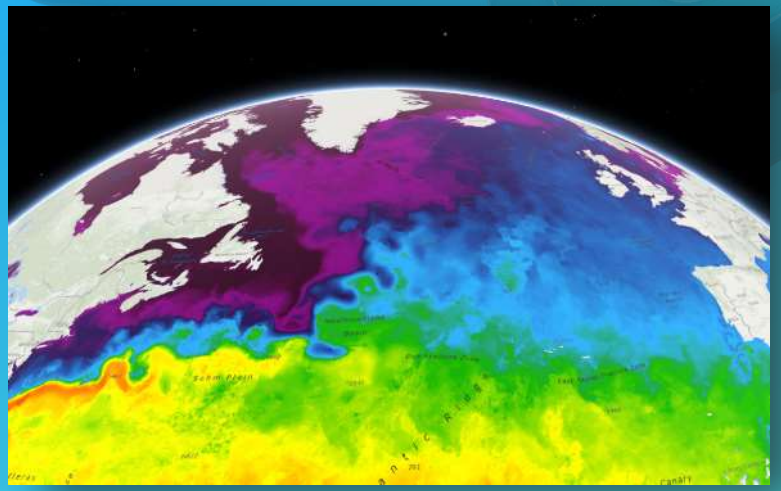
- Provides visual mapping component & widgets
- Support for many different layer types (data sources)
- Integration with ArcGIS platform (security, sign-in, premium services, …)

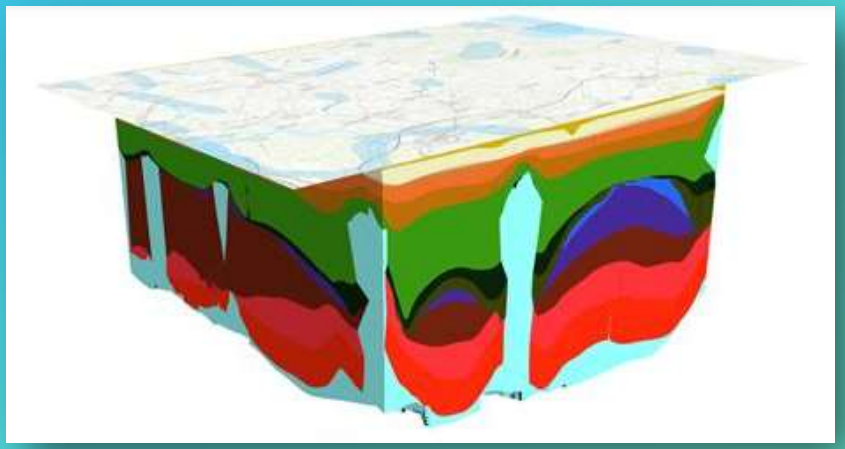
- Get it today
  - Hosted build https://js.arcgis.com/4.6
  - Doc/samples https://developers.arcgis.com/javascript



Introducing 3D

- Data
  - 2D tiles, maps, elevation, features
  - Scene layers (open i3s format)
    - 3D Objects, Integrated Meshes, Point Clouds
- 3D concepts
  - Local & global scenes
  - Ground surface with elevation
  - Camera, light and shadows
- Requirements
  - Modern web browser w/ WebGL (IE11+)
  - Modern hardware w/ Graphics Card





Modern, simple API

- **Properties**
  https://developers.arcgis.com/javascript/latest/guide/working-with-props/index.html
  - read/write properties directly
  - set all properties via constructors
  - watch properties, instead of events

```
// Creates a new Map with a 'streets' basemap
var map = new Map({
  basemap: 'streets'
});

// Read the basemap property
console.log("Basemap title: ", map.basemap.title);
```

- **Promises**
  https://developers.arcgis.com/javascript/latest/guide/working-with-promises/index.html
  - for handling asynchronous tasks, e.g. network
  - states: *pending*, *resolved*, or *rejected*

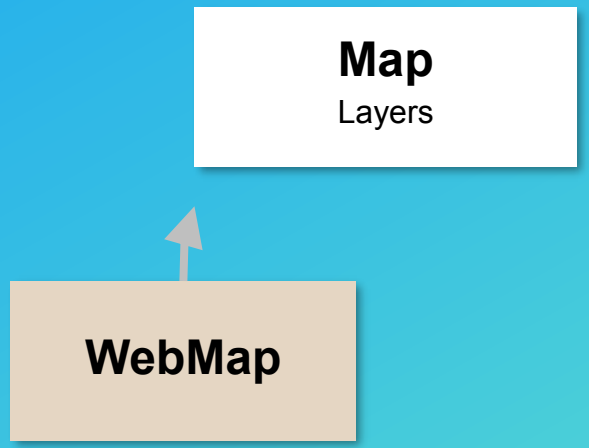
```
someAsyncFunction().then(function(resolvedVal){
  // This is called when the promise resolves
  console.log(resolvedVal);
}, function(error){
  // This function is called when the promise is rejected
  console.error(error);
});
```

- Autocasting, Loadable, JSON, Typescript, …

ArcGIS API for JavaScript

Build 3D web apps!

3.x architecture



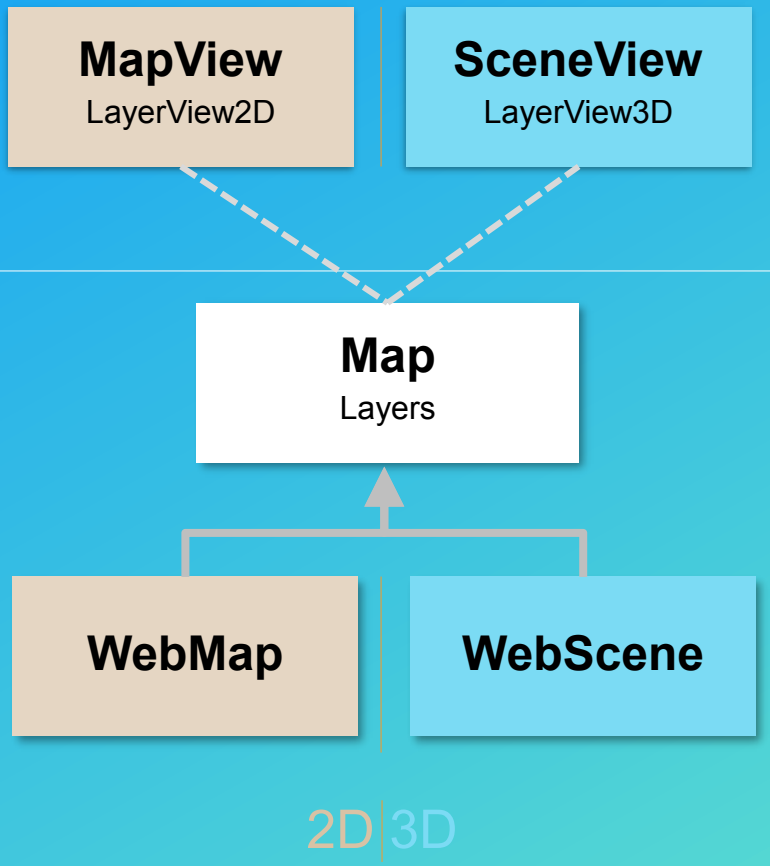
4.x architecture

Renders and interacts with the model

View

Model

Describes the content of the map/scene



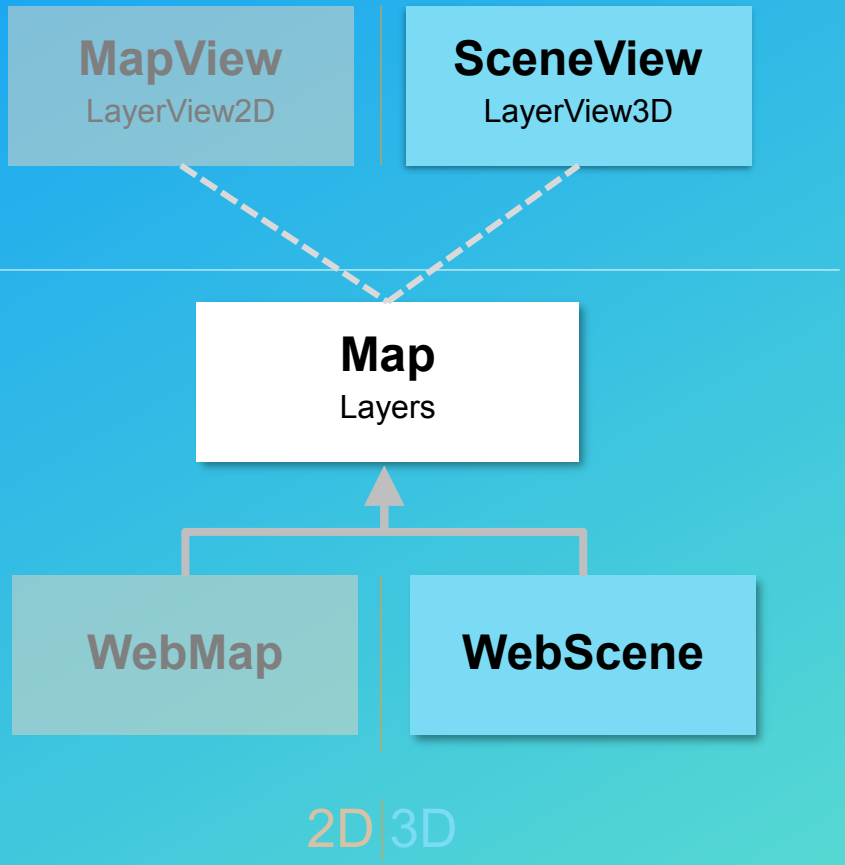
4.x architecture – 3D

Renders and interacts with the model

View

Model

Describes the content of the map/scene



API Concepts

Working with the `SceneView`

Start with a 2D map/view, replace `MapView` with `SceneView`

```
view = new MapView({
  container: "viewDiv",

  map: new Map({
    basemap: "streets",

    layers: [new TileLayer(
      "...New_York_Housing_Density/MapServer"
    )]
  })
});
```

```
view = new SceneView({
  container: "viewDiv",

  map: new Map({
    basemap: "streets",

    layers: [new TileLayer(
      "...New_York_Housing_Density/MapServer"
    )]
  })
});
```

Working with the `SceneView`

- Unified 2D and 3D data model (the `Map`), `Layer`, `Renderer` and `Symbol`
- Common `View` subclass between 2D and 3D
- Makes it easy (boring!) to transition to 3D
- However, to exploit 3D, you need to learn certain 3D concepts

Working with the `SceneView`

```
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`

```
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;
}
```

Camera

```
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;
```

goTo

```
const h = view.camera.heading;

// Set the heading of the view to
// the closest multiple of 30 degrees
const heading = Math.floor(h / 30) * 30 + 30;

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

```
const h = view.camera.heading;

// Set the heading of the view to
// the closest multiple of 120 degrees
const heading = Math.floor(h / 120) * 120 + 120;

const target = {
  heading: heading
};

const options = {
  easing: "out-cubic",
  speedFactor: 0.2
};

view.goTo(target, options);
```

toMap, toScreen and hitTest

```
view = new SceneView({
  map: new Map({
    basemap: "satellite"
  })
});

for (var x = 1; x <= 2; x++) {
  for (var y = 1; y <= 2; y++) {
    var px = x * (view.width / 3);
    var py = y * (view.height / 3);

    view.graphics.add(new Graphic({
      geometry: view.toMap(px, py),
      symbol: symbol,
      attributes: { x: x, y: y }
    }));
  }
}
```

```
view.on("pointer-move", event => {
  view.hitTest({ x: event.x, y: event.y })
    .then(result => {
      var graphic = result.results[0] &&
          result.results[0].graphic;

      var attrs = graphic && graphic.attributes;

      if (attrs) {
        logDiv.textContent = `${attrs.x}/${attrs.y}`;
      }
    });
});
```

Map

- A Map can be used with both `MapView` and `SceneView`
- `Basemap` is exactly the same concept as in 2D
- `Ground`: defines the ground surface of the scene
  - Elevation layers
  - Default world elevation layer (`"world-elevation"`)
  - Surface properties (coming later)

Layers

Layer type	Characteristics
`FeatureLayer`	vector
`CSVLayer`	vector
`StreamLayer`	vector
`MapImageLayer`	dynamic
`ImageryLayer`	dynamic
`WMSLayer`	dynamic
`OpenStreetMapLayer`	raster, cached
`TileLayer`	raster, cached
`WebTileLayer`	raster, cached
`WMTSLayer`	raster, cached
`VectorTileLayer`	vector, cached

Layers 3D — `elevationInfo`

- `relative-to-scene` (new)
- `relative-to-ground`, `absolute-height`, `on-the-ground`

```
layer1.elevationInfo = {
  mode: "on-the-ground"
};

layer2.elevationInfo = {
  mode: "relative-to-ground",
  offset: 3000
};

layer3.elevationInfo = {
  mode: "absolute-height",
  offset: 3000
};
```

Layers 3D

Layer type	Characteristics
`ElevationLayer`	raster, cached
`SceneLayer`	vector
`IntegratedMeshLayer`	vector
`PointCloudLayer`	vector

Layers 3D — `ElevationLayer`

- Elevation services
  - Tiled image service
  - LERC format (Limited Error Raster Compression)
- Added to `map.ground.layers`
  - Multiple elevation layers overlay just as map layers do
- `async queryElevation(geometry)`
- `async createElevationSampler(extent)`

Layers 3D — ElevationLayer

```
var map = new Map({
  basemap: "satellite",
  ground: "world-elevation" // use default world elevation layer
});

// load an additional elevation layer depicting the
// elevation difference after the 2014 Oso landslide
var elevationLayer = new ElevationLayer({
  url: ".../OsoLandslide_After_3DTerrain/ImageServer",
  visible: false
});

map.ground.layers.add(elevationLayer);

var view = new SceneView(...);
```

```
// toggle the elevation layer visiblity
elevationLayer.visible = !elevationLayer.visible
```

```
// query elevation at the clicked map point
view.on("click", function(event) {
  var position = event.mapPoint;
  var query = elevationLayer.queryElevation(position);
  query.then((result) => {
    console.log(result.geometry.z)
  });
});
```

Layers — `SceneLayer` (points)

```
// all the typical initialization
var map = new Map(...);
var view = new SceneView(...);

// Create SceneLayer and add to the map
var sceneLayer = new SceneLayer({
   url: ".../Airports_PointSceneLayer/SceneServer"
});

map.add(sceneLayer);
```

Layers — `SceneLayer` (3d object)

```
// all the typical initialization
var map = new Map(...);
var view = new SceneView(...);

// Create SceneLayer and add to the map
var sceneLayer = new SceneLayer({
  url: ".../NYCatt/SceneServer",
});

map.add(sceneLayer);
```

Layers — SceneLayer (3d object)

```
// all the typical initialization
var map = new Map(...);
var view = new SceneView(...);

// Create SceneLayer and add to the map
var sceneLayer = new SceneLayer({
  url: ".../NYCatt/SceneServer",
});

map.add(sceneLayer);
```

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

```
// reset filter
sceneLayer.definitionExpression = null;
```

```
// filter out tall buildings
sceneLayer.definitionExpression =
  "HEIGHTROOF < 100 AND HEIGHTROOF > 0";
```

Layers — IntegratedMeshLayer

```
// create the integrated mesh layer
var layer = new IntegratedMeshLayer({
  url: ".../Oxford_Scene/SceneServer"
});

// create a map with the layer added
var map = new Map({
  basemap: "streets",
  layers: [layer],
  ground: "world-elevation"
});

// finally, create a view with a good
// perspective on the integrated mesh
var view = new SceneView({
  container: "viewDiv",
  map: map,
  camera: {
    position: [-1.2567, 51.7517, 123.6058],
    heading: 41.4698,
    tilt: 75.9609
  }
});
```

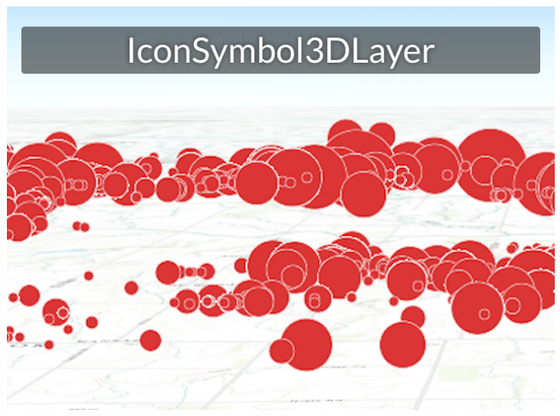
Layers — PointCloudLayer

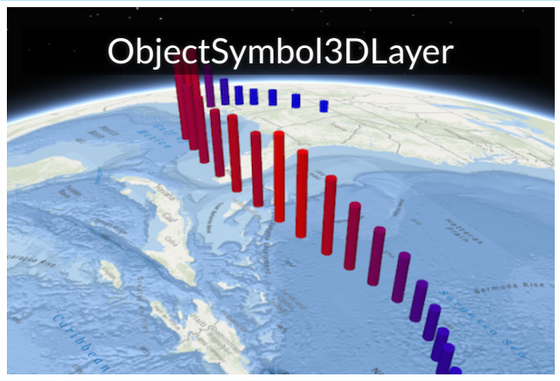
```
// all the typical initialization
var map = new Map(...);
var view = new SceneView(...);

// create Point Cloud Layer
var pcLayer = new PointCloudLayer({
  url: ".../BARNEGAT_BAY_LiDAR_UTM/SceneServer"
});

map.add(pcLayer);
```

```
// update the render settings
function updateRenderer(pointSize, density) {
  pcLayer.renderer = new PointCloudRGBRenderer({
    field: "RGB",
    pointSizeAlgorithm: {
      type: "fixed-size",
      useRealWorldSymbolSizes: false,
      size: pSize
    },
    pointsPerInch: density
  })
}
```

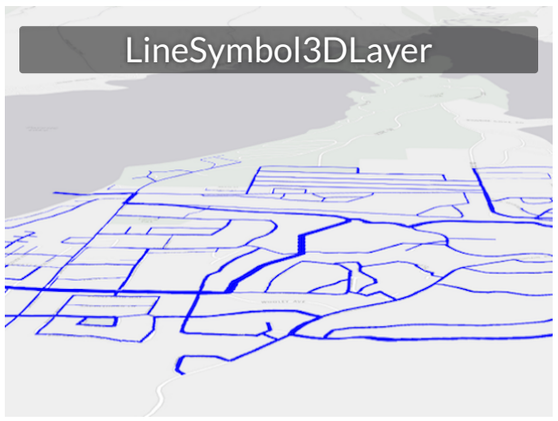
Symbology

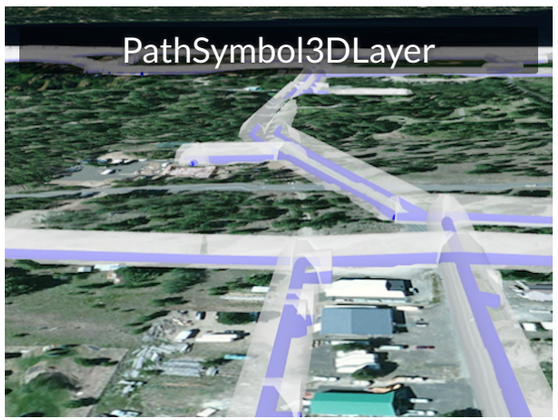
point



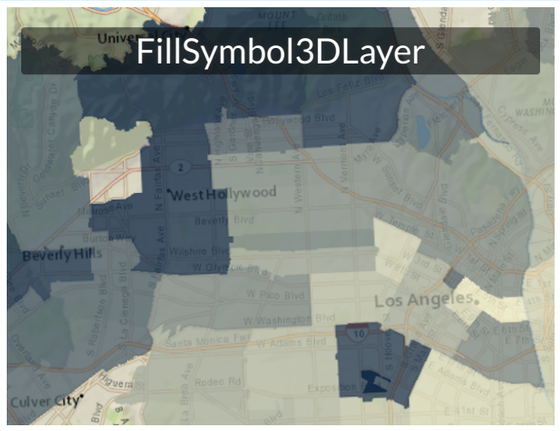


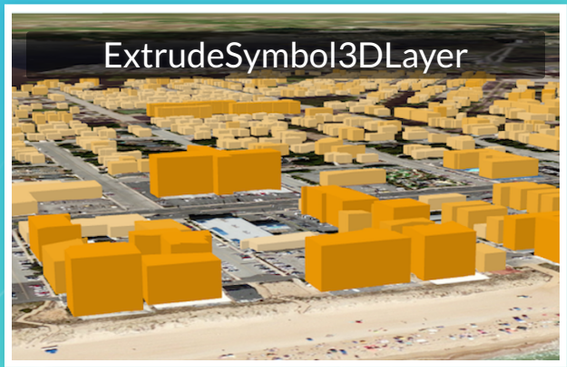
line





polygon





Symbology — `Renderer` example

- Visual variables and SceneLayer sample

Symbology — `WebStyleSymbol`


```
const webStyleSymbol = new WebStyleSymbol({
  name: "Traffic_Barrier_1",
  styleName: "EsriRealisticStreetSceneStyle"
});
```

Example in symbol playground

WebScene

WebScene

- Defines the content and style of a 3D Scene
- Serialized as JSON and stored in Portal/Online

WebScene — Spec

- JSON spec similar to WebMap
- Layers, basemap, slides, initial state (position and light)
- Metadata: scene type, spatial reference, version...
- https://developers.arcgis.com/web-scene-specification/

WebScene — API

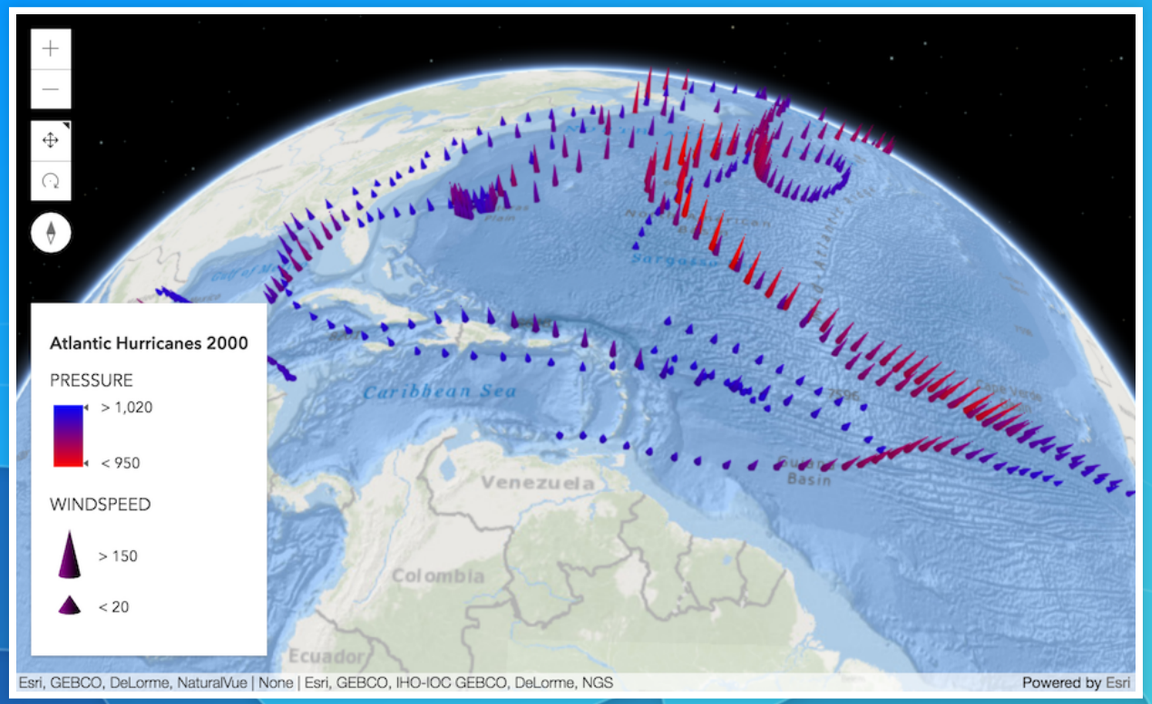
- Supported by `SceneView`
- Loadable from `Portal`

```
class WebScene extends Map {
  presentation: {
    slides: Collection<Slide>;
  };

  initialViewProperties: {
    viewpoint: Viewpoint;
    environment: Environment;
    spatialReference: SpatialReference;
    viewingMode: "global" | "local";
  };

  portalItem: PortalItem;

  clippingArea: Extent;
  clippingEnabled: boolean;
}
```

WebScene — `viewingMode`

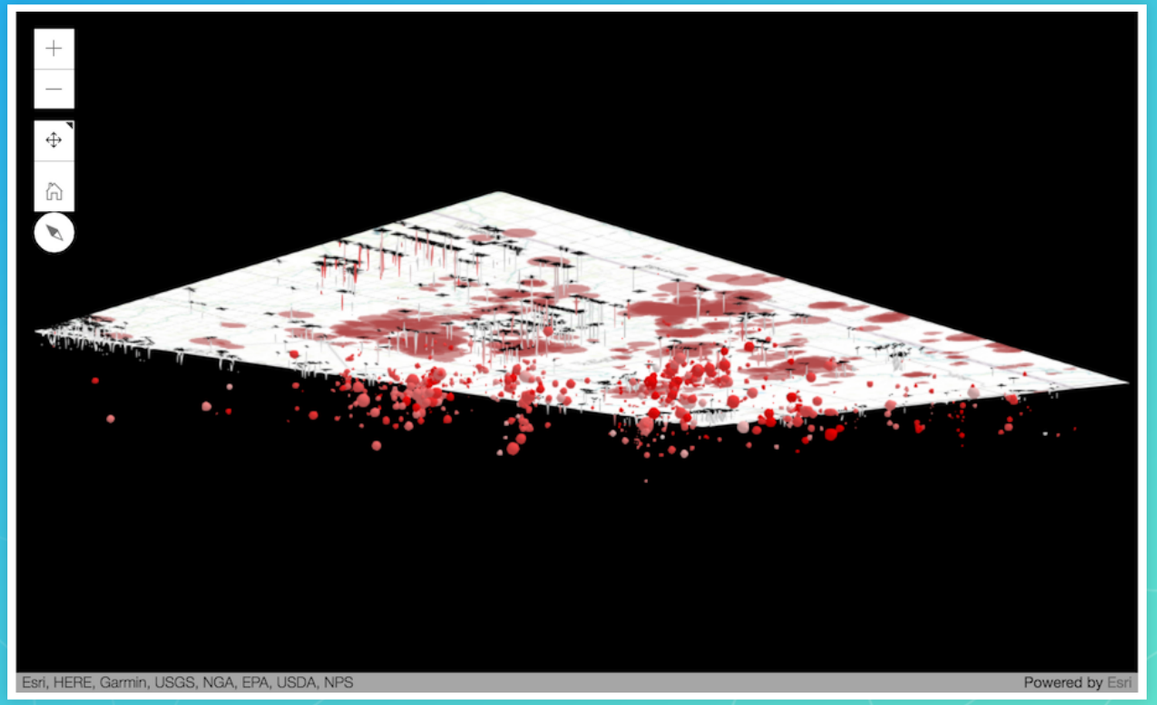
Global

geographic, global extent, spherical



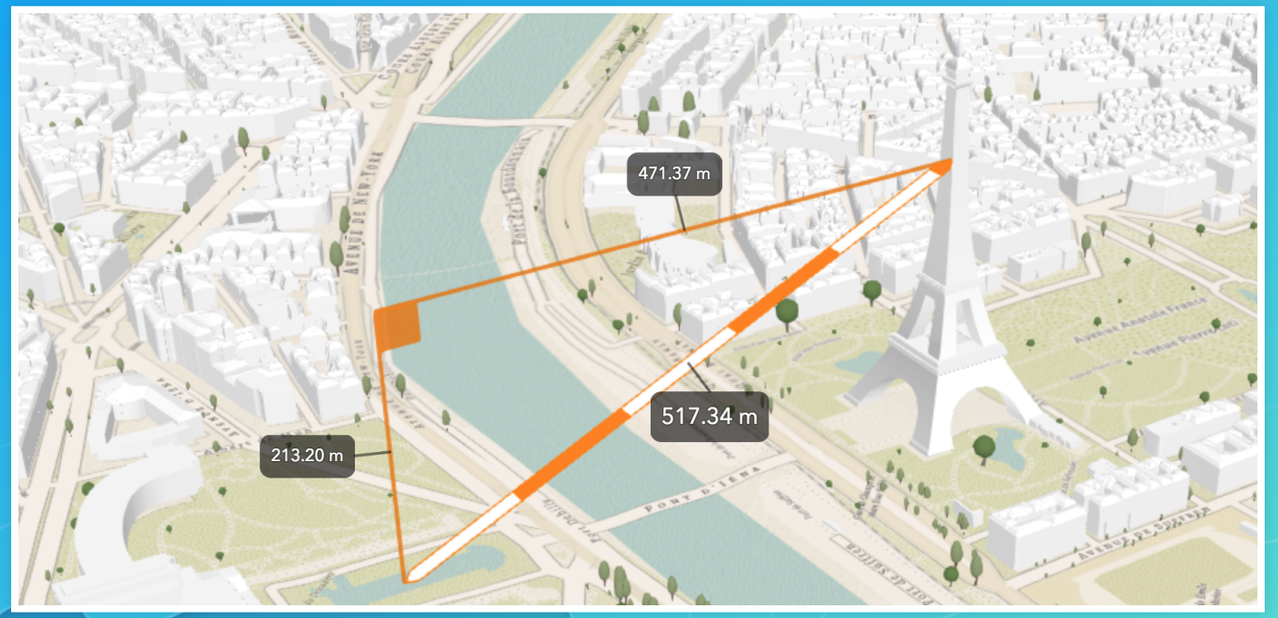
Local

projected, local extent, planar, clipping

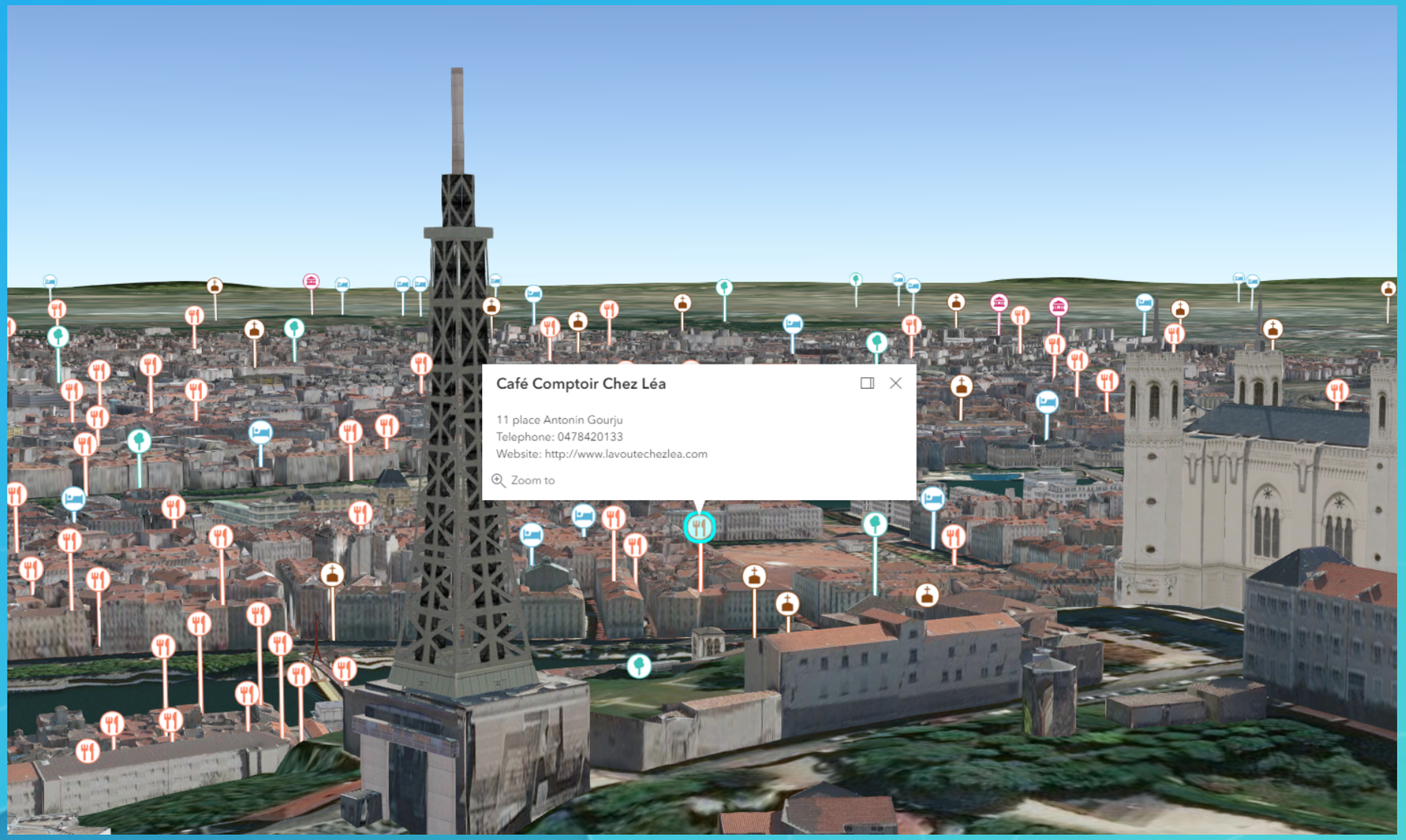


New 3D widgets

- Direct line measurement
- `esri/widgets/DirectLineMeasurement3D`



Building an App


Café Comptoir Chez Léa
11 place Antonin Gourju
Telephone: 0478420133
Website: http://www.lavoutechezlea.com
Zoom to

Building an App: Lyon

```
var renderer = new UniqueValueRenderer({
  field: "Type",
  uniqueValueInfos: [{
    value: "Museum",
    symbol: createSymbol("Museum.png", "#D13470")
  }, {
    value: "Restaurant",
    symbol: createSymbol("Restaurant.png", "#F97C5A")
  }, {
    value: "Church",
    symbol: createSymbol("Church.png", "#884614")
  }, {
    value: "Hotel",
    symbol: createSymbol("Hotel.png", "#56B2D6")
  }, {
    value: "Park",
    symbol: createSymbol("Park.png", "#40C2B4")
  }]
});

var featLayer = new FeatureLayer({
  portalItem: {
    id: "5acdabddfb1f4852932d43f51fca57eb"
  },
  renderer: renderer
});
```

Elevation mode: `relative-to-scene`

```
var featLayer = new FeatureLayer({
  portalItem: {
    id: "5acdabddfb1f4852932d43f51fca57eb"
  },
  renderer: renderer,
  elevationInfo: {
    mode: "relative-to-scene"
  }
});
```

Declutter

Number of overlapping points is reduced

```
var featLayer = new FeatureLayer({
  portalItem: {
    id: "5acdabddfb1f4852932d43f51fca57eb"
  },

  renderer: renderer,

  elevationInfo: {
    mode: "relative-to-scene"
  },

  featureReduction: {
    type: "selection"
  }
});
```

Improve Perspective

Improve the sense of depth for 2D icons

```
var featLayer = new FeatureLayer({
  portalItem: {
    id: "5acdabddfb1f4852932d43f51fca57eb"
  },

  renderer: renderer,

  elevationInfo: {
    mode: "relative-to-scene"
  },

  featureReduction: {
    type: "selection"
  },

  screenSizePerspectiveEnabled: true
});
```

Callout Lines

Lift symbols above the ground for better visibility.

```
new PointSymbol3D({
  symbolLayers: ...,

  verticalOffset: {
    screenLength: 30,
    maxWorldLength: 200
  },

  callout: new LineCallout3D({
    color: "white",
    size: 2,

    border: {
      color: color
    }
  })
});
```

3D Models

Use WebStyleSymbols to include 3d models in your scene

```
new UniqueValueRenderer({
  field: "TYPE",
  uniqueValueInfos: [
    {
      value: "taxi",

      symbol: new WebStyleSymbol({
        name: "Taxi",
        styleName: "EsriRealisticTransportationStyle"
      })
    }
  // ...
  ]
});
```

Webscene loading

Loading a webscene by id

```
var map = new WebScene({
  portalItem: {
    id: "8b464fd68d87480486d36e8cbbc52ab9"
  }
});

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

Popups and webscene saving

Adding a custom popup template and saving the webscene

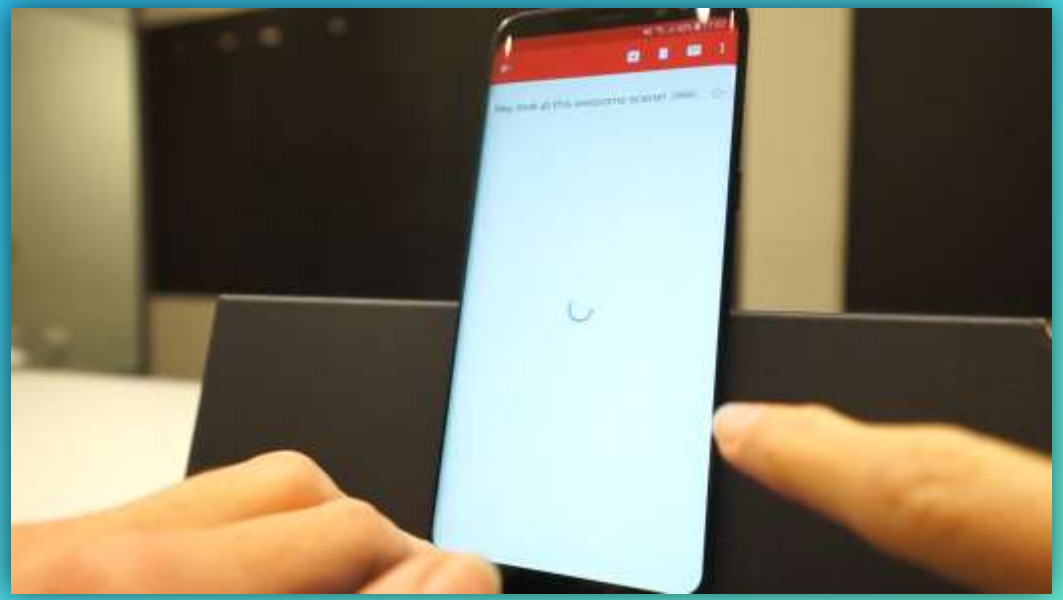
```
featLayer.popupTemplate = new PopupTemplate({
    title: "{NAME}",
    content: "{ADDRESS}<br>" +
    "Telephone: {TELEPHONE}<br>" +
    "Website: {WEBSITE}<br>"
  });
});

// ...

var portal = new Portal({
  url: myPortalUrl,
  authMode: "immediate"
});

portal.load().then(function() {
  webscene.saveAs({
    title: "My Scene",
    portal: portal
  });
});
```

What's new

What's new in 4.7

- Mobile support
  - iOS → iPhone 8, iPad Pro (Safari browser)
  - Android → Samsung S8, Samsung Tab S3 (Chrome browser)
  - More devices in the future

→ Works in modern phones and tables
  - Hardware Requirements
    - 2+ GB of RAM (4 GB recommended)
    - Good GPU



What's new in 4.7

- **3D Area Measurement tool**

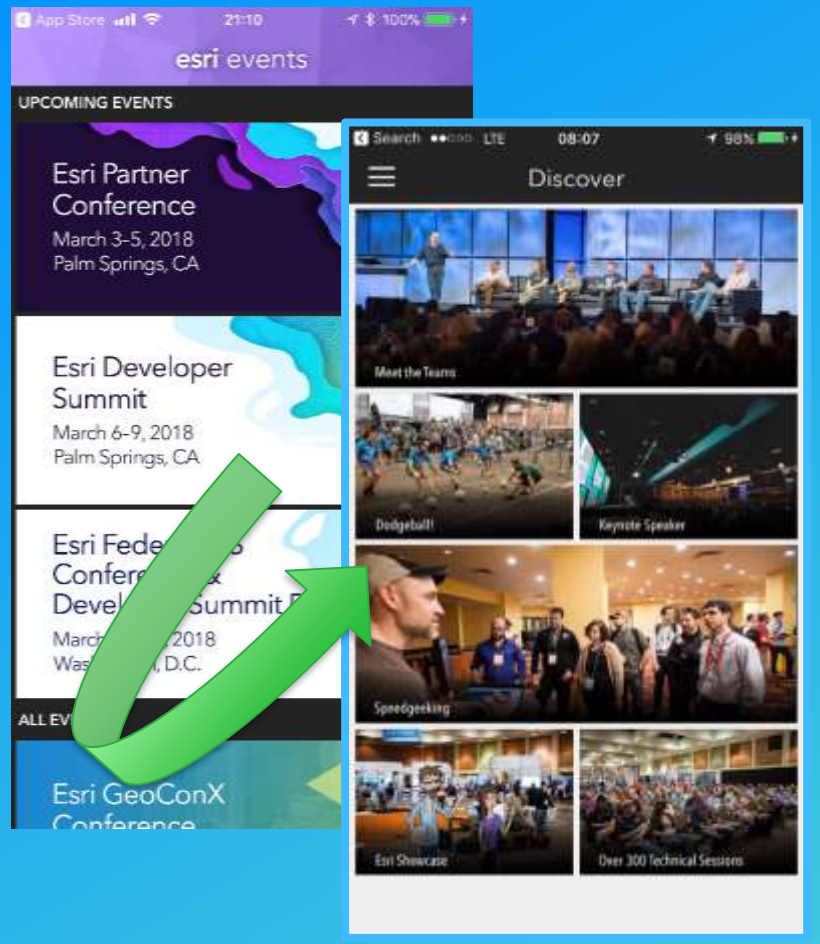
- **Edge Rendering**
  - Session: 3D Visualization with the ArcGIS API for JavaScript
  - Thursday, March 08 | 10:30 am - 11:30 am | Smoketree A-E

- **Mesh Geometry API**
  - Session: Advanced 3D Features in the ArcGIS API for JavaScript
  - Thursday, March 08 | 5:30 pm - 6:30 pm | Smoketree A-E

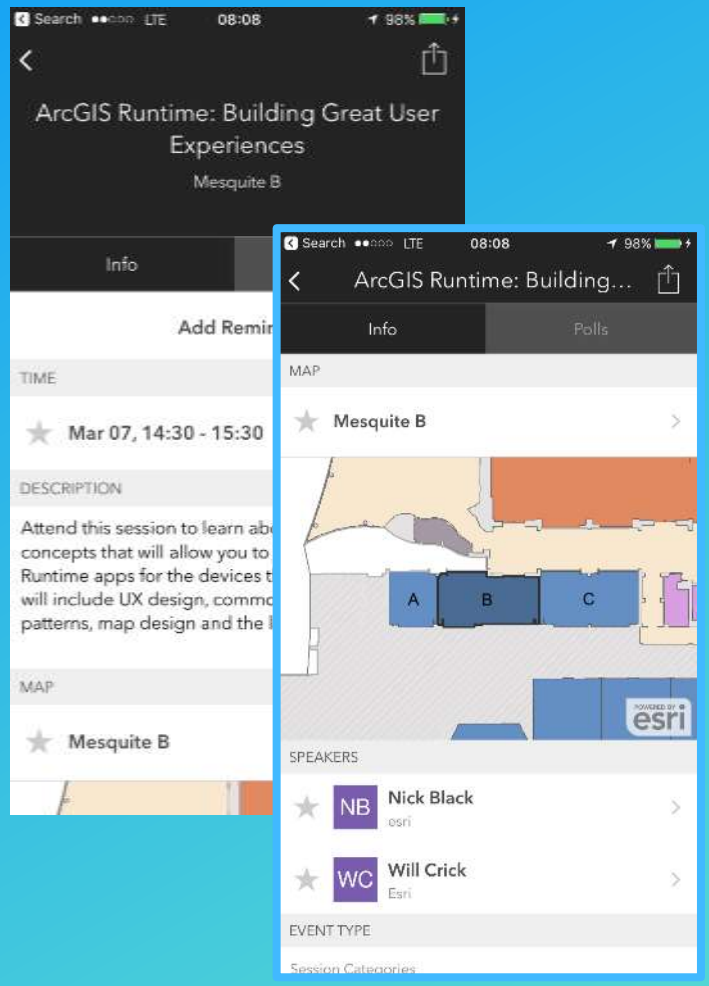
Thank you

Please Take Our Survey!

Download the Esri Events app and find your event

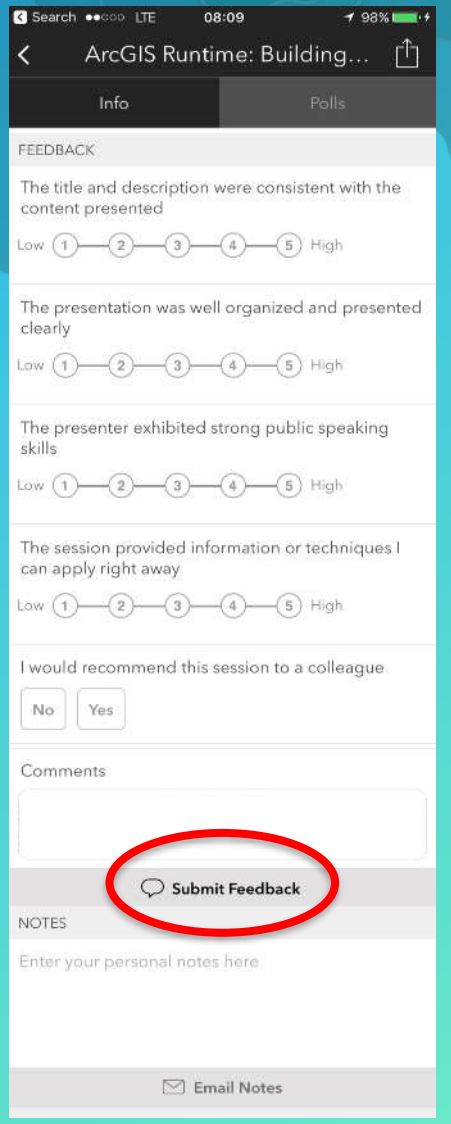


Select the session you attended



Scroll down to the "Feedback" section

Complete Answers, add a Comment, and Select "Submit"




esri
THE
SCIENCE
OF
WHERE