Building 3D GIS applications for the browser using JavaScript

Janett Baresel and Pascal Mueller

ESRI R&D Center Zürich

Table of contents

Urban planning application

API basics

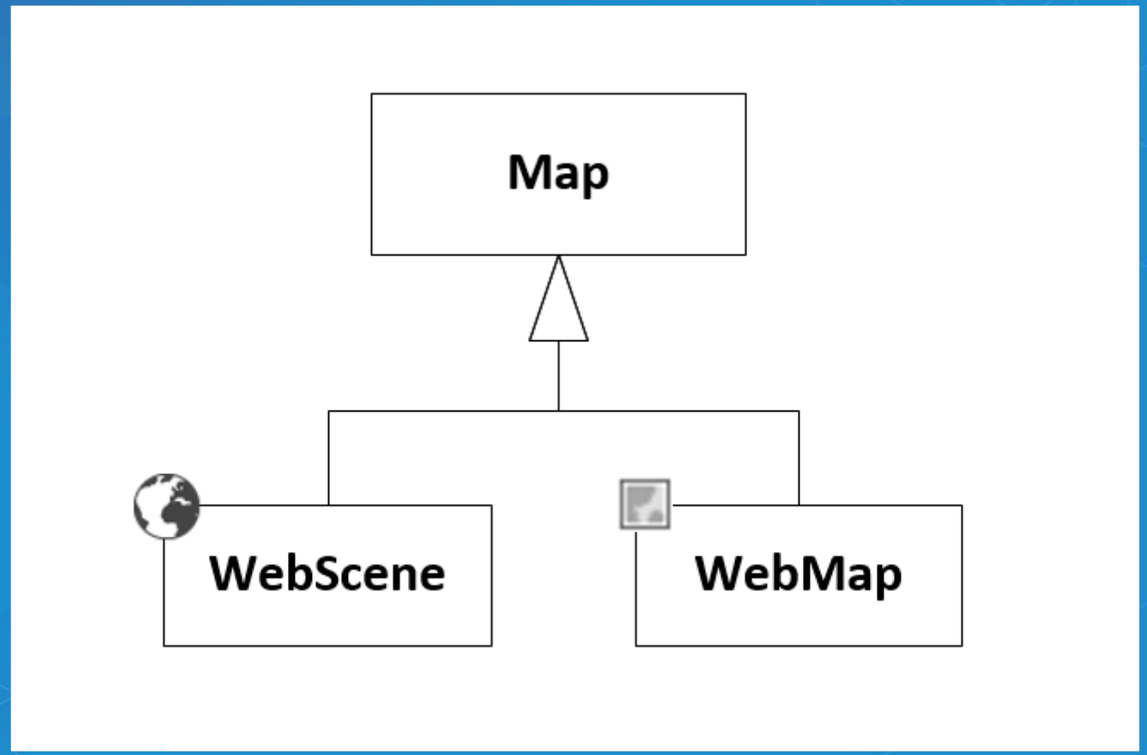
Create a new scene

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


var view = new SceneView({
   container: "viewDiv",
   map: map,
   scale: 100000000,
   center: [-101.17, 21.78]
});
```

API basics

The WebScene



API basics

Open a scene

```
var webscene = new WebScene({
  portalItem: {
    id: "33a2a791a22b4cf1b23858f463363bfe"
  }
});

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

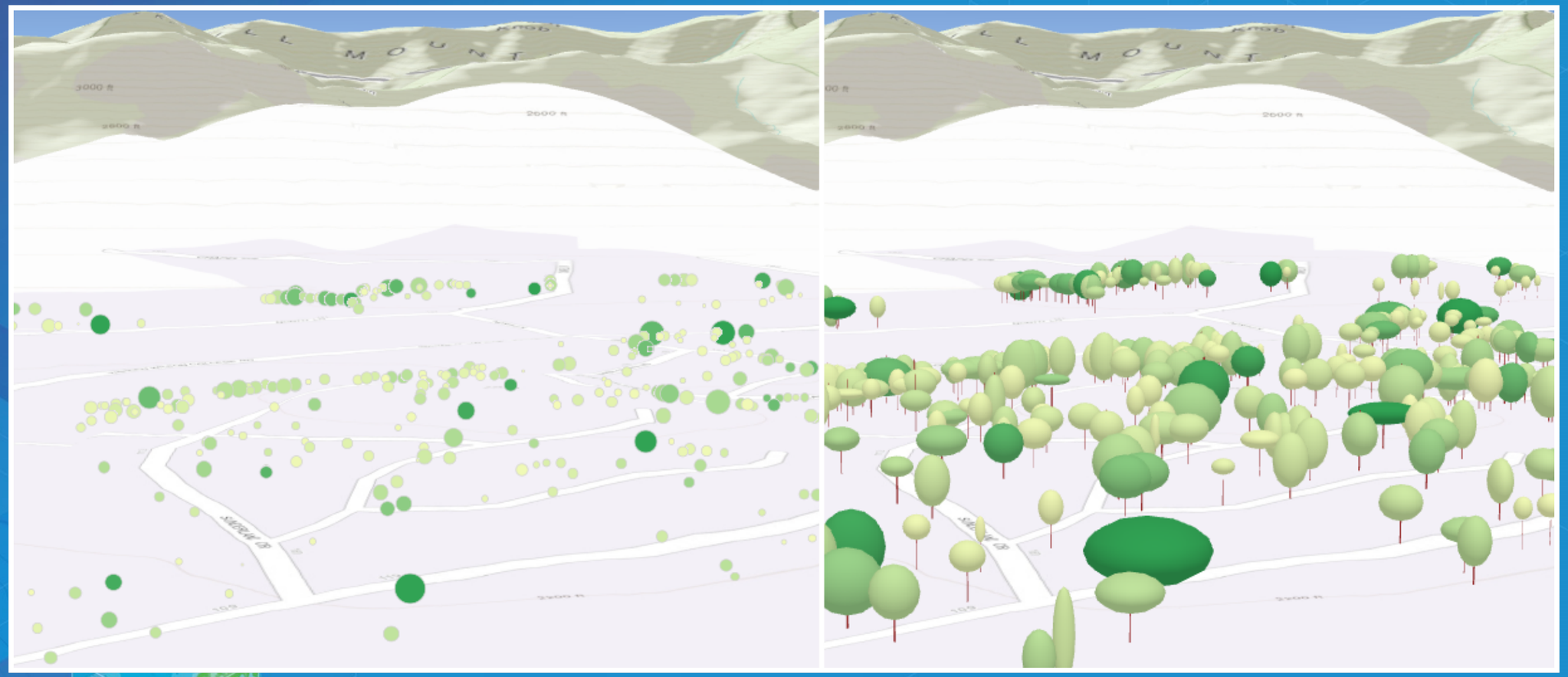
API basics

Add a layer

```
var layer = Layer.fromPortalItem({
        portalItem: {
          id: "ae411243d9814a3784d3ba1f90943390"
        }
      }).then(function(layer){
        webscene.add(layer);
    });
```

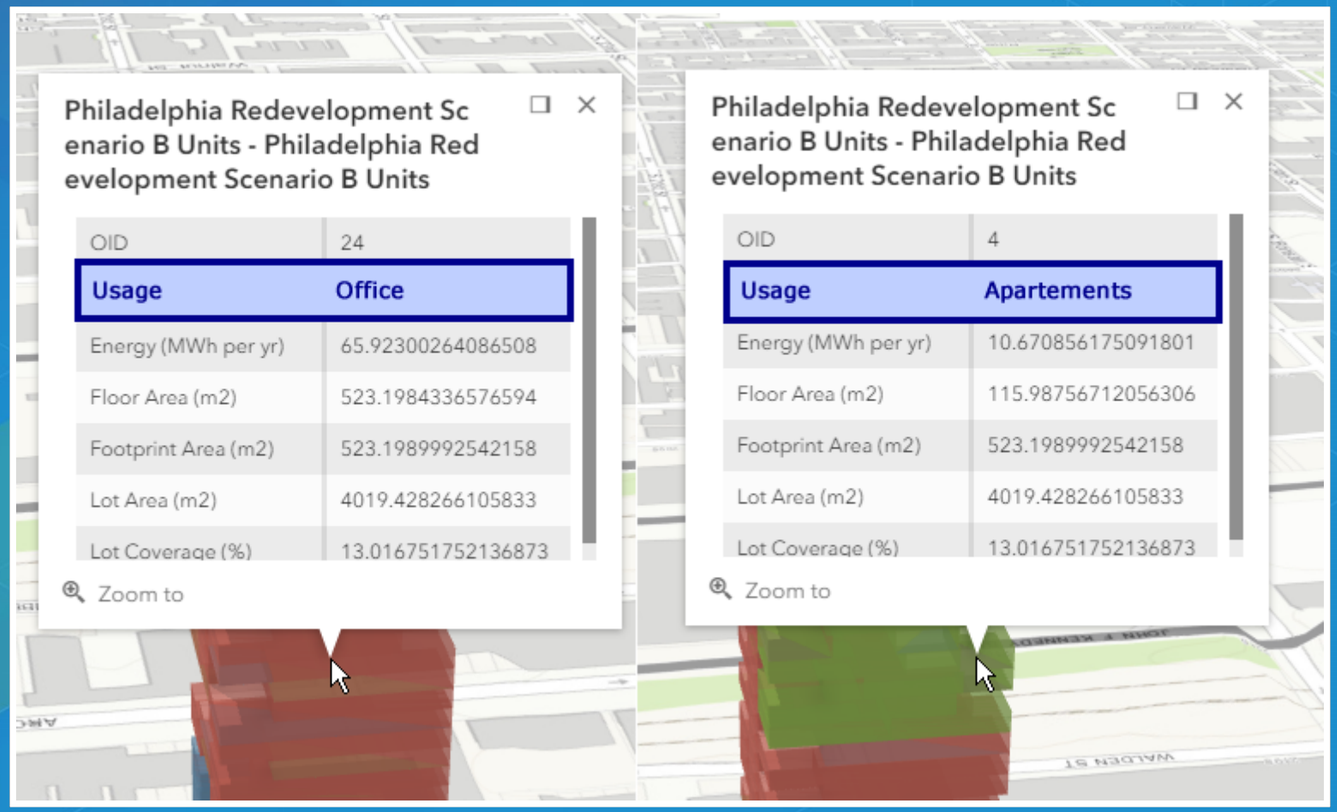
Attribute-driven visualization

- **Renderers define how to visually represent each feature in a Layer.**



Attribute-driven visualization

- **UniqueValueRenderer** allows you to symbolize features in a FeatureLayer and SceneLayer based on one or more matching string attributes.



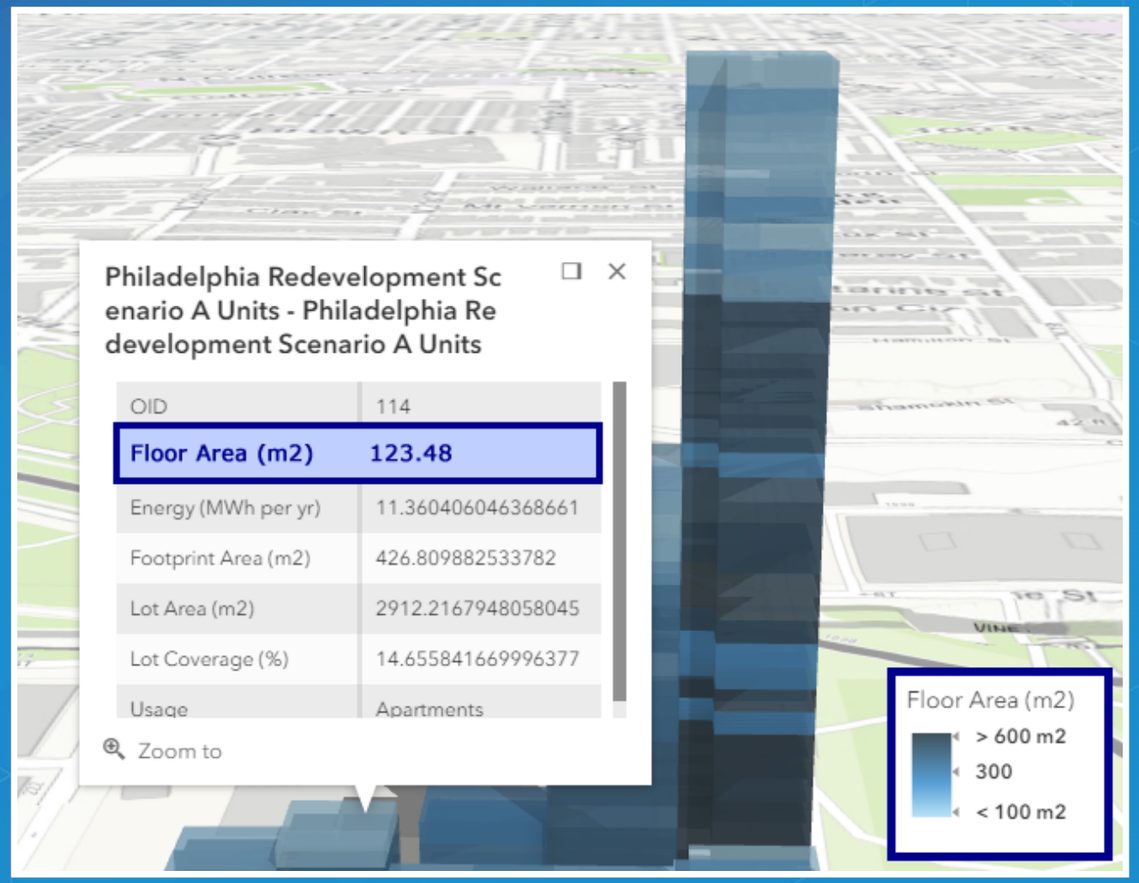
Attribute-driven visualization

Unique strings with the UniqueValueRenderer

```
var usageRenderer = new UniqueValueRenderer({
   field: "Usage",
   uniqueValueInfos: [
     {
       value: "Office",
       symbol: new MeshSymbol3D({
         symbolLayers: [new FillSymbol3DLayer({
           material: {color: "#fd7f6f"}
         })]
       })
     },
     {
       value: "Hotel",
       symbol: new MeshSymbol3D({
         symbolLayers: [new FillSymbol3DLayer({
           material: {color: "#7eb0d5"}
         })]
       })
     },
     ...
   ]
   });
layer.renderer = usageRenderer;
```

Attribute-driven visualization

A range of values with the SimpleRenderer and VisualVariables



Attribute-driven visualization

```
var energyRenderer = new SimpleRenderer({
    symbol: new MeshSymbol3D({
      symbolLayers: [new FillSymbol3DLayer({
        material: {color: "#b8e4f8"}
      })]
    }),
    visualVariables: [{
      type: "color",
      field: "Energy (MWh/yr)",
      stops: [
        {
          value: 1
          color: "#43a94d",
        },
        {
          value: 100,
          color: "#fbfdbb"
        },
        {
          value: 200,
          color: "#db2a24"
        }]
    }]
  });
layer.renderer = energyRenderer;
```

Legend Widget*

```
var legend = new Legend({
  view: view
});

view.ui.add(legend, "bottom-right");
```

* JS API 4.1 will support legends for scene layers and many more.

Popups

```
layer.popupTemplate = new PopupTemplate({
  title: layer.title,
  content: "{*}",
  fieldInfos: layer.fields && layer.fields.map(function (field) {
    return {
      fieldName: field.name,
      label: field.alias,
      visible: true
    };
  })
});
```

Time Slider

```
view.environment.lighting.directShadowsEnabled = true;
view.environment.lighting.ambientOcclusionEnabled = true;

function timeSliderChanged(value){
  var hours = calculateHoursFromValue(value);
  var minutes = calculateMinutesFromValue(value);
  var newDate = new Date(view.environment.lighting.date.getTime());
  newDate.setUTCHours(hours);
  newDate.setUTCMinutes(minutes);
  view.environment.lighting.date = newDate;
}

<input type="range" oninput="timeSliderChanged(value)">
```

Split View

- 2 views with synchronized cameras
- be aware of the performance implications

```
var view1 = new SceneView({
  container: "view1Div",
  map: webScene
});

var view2 = new SceneView({
  container: "view2Div",
  map: webScene
});
```

Split View

- synchronization of the camera

```
all([view1, view2]).then(function() {
  view1.watch("camera", function(camera) {
    if (view1.interacting || view1.animation) {
      view2.camera = camera;
    }
  });
  view2.watch("camera", function(camera) {
    if (view2.interacting || view2.animation) {
      view1.camera = camera;
    }
  });
});

view1.on('layerview-create', function(evt) {
  if (evt.layer === theLayerToNotShowHere) {
    evt.layerView.visible = false;
  }
});
```

More API: Presentation

- slide contains: viewpoint, layer visibility, basemap, environment + metadata

```
webscene.presentation = {
  slides : []
}

// capture current scene state
var slide = Slide.createFrom(view);

// re-apply the stored state
slide.applyTo(view);
```

More API: Camera

```
new Camera({
    position: new Point({
      x: -116.54, // longitude
      y: 33.83, // latitude
      z: 1000, // altitude in meters
      spatialReference: SpatialReference.WGS84
    }),
    heading: 30, // 0 .. 360°, clockwise
    tilt: 45 // 0..180°, 0° straight down
  })
```

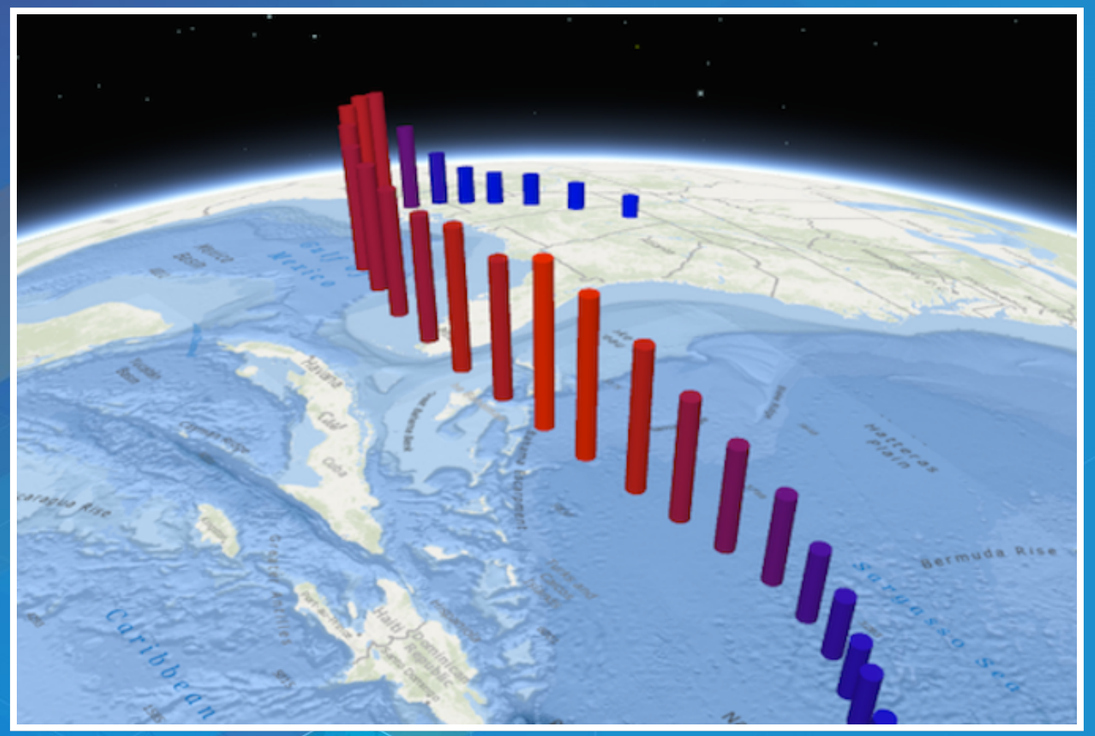
More API: Animation

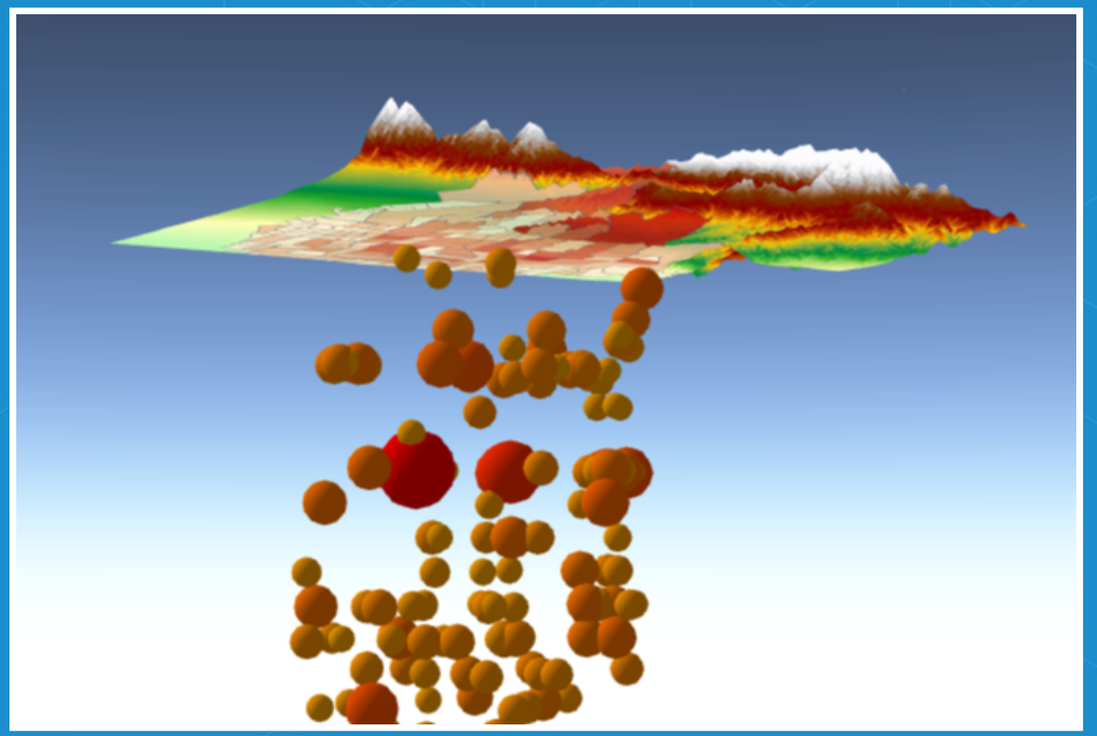
```
view.goTo(target, options);
```

- Target can be
  - [longitude, latitude] pair of coordinates
  - Geometry (or array of Geometry[])
  - Graphic (or array of Graphic[])
  - Viewpoint
  - Camera

More API: Global - local

Global	Local
geographic, global extent, spherical	projected, local extent, planar





More API: Global - local

```
scene = new WebScene({
  viewingMode: 'local',
});
```

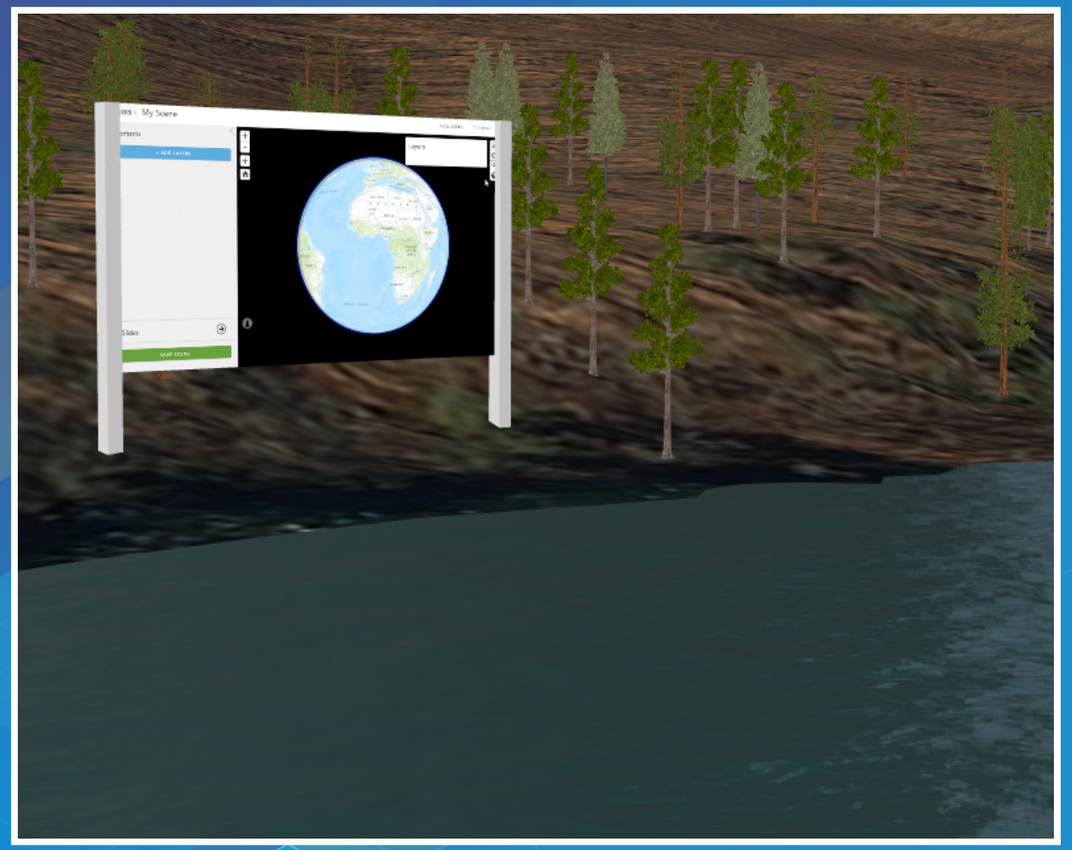
```
scene = new WebScene({
  viewingMode: 'global',
});
```

More API: Clipping

- clipping is only supported for local scenes

```
scene = new WebScene({
  viewingMode: 'local',
  clippingArea: {
    xmin: 344556.17949990794,
    ymin: 3786680.957522931,
    xmax: 368905.9689491527,
    ymax: 3801033.594521225,
    spatialReference: { wkid: 26711 }
  },
  clippingEnabled: true
});
```

More API: External Rendering




Schematic
Usage
Office
Hotel
Apartments
Retail
Textured
Usage
Area
Energy Consumption
Time of the Day
RE, DeLorme, Intermap, increment P Corp., GEBCO, USGS, FAO, NPS, NRC
eLorme, Intermap, increment P Corp., GEBCO, USGS, FAO, NPS, NRCAN...
Powered by Esri