Advanced 3D Features

ArcGIS API for JavaScript

Jesse van den Kieboom, ESRI R&D Center Zürich

Thomas Other, ESRI R&D Center Zürich

Live version of this presentation is available on:
https://jkieboom.github.io/devsummit-palm-springs-2018/presentations/advanced-3d-arcgis-js-api

Agenda

1. Introduction to 3D
   - 4.x Foundations
   - Working with the 3D SceneView
   - Adding custom graphics
2. Customized 3D Visualizations
   - Custom Mesh
   - External Renderer
   - Camera fly-by

ArcGIS API for JavaScript

4.x Foundations

JavaScript API

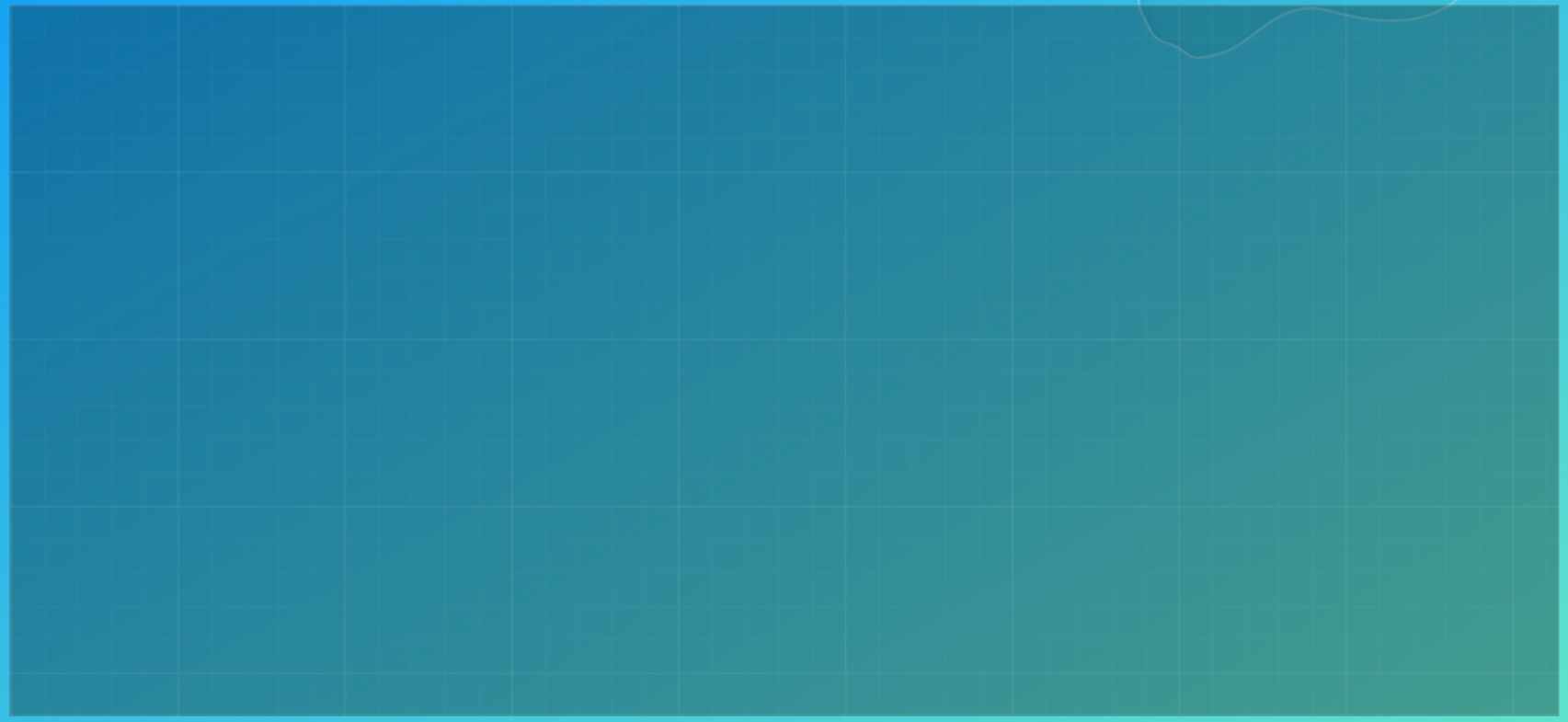
Basic Concepts

- Unified 2D and 3D `Map` data model:
  - `Layer` — Fundamental Map component
  - `Renderer` — Visualization Methods
  - `Symbol` — Symbolization Instructions
- Common `View` subclass between 2D and 3D:
  - `MapView` — 2D Visualization
  - `SceneView` — 3D Visualization
- Makes it easy to transition between 2D and 3D
- To use 3D effectively one needs to understand 3D concepts

JavaScript API

2D & 3D Viewing

```
var map = new Map({
  basemap: "streets",

  layers: [new FeatureLayer(
    "...Germany/FeatureServer/0"
  )]
});

viewLeft = new MapView({
  container: "viewDivLeft",

  map: map
});

viewRight = new SceneView({
  container: "viewDivRight",

  map: map
});
```



JavaScript API

Promises

- All asynchronous operations are modeled as a `Promise`
- Certain classes *are* promises themselves (`SceneView` , `MapView` , `Layer`)
- `Promises` are chainable and allow writing sequential asynchronous code

```
view
    .when(() => {
      // View is ready to be interacted with, load the layer
      return layer.load();
    })

    .then(() => {
      // Layer is now loaded, project extent using geometry service
      return geometryService.project([layer.fullExtent]);
    })

    .then((projected) => {
      // Extent has been projected, we can now go to it
      return view.goTo(projected[0]);
    })

    .then(() => {
      // Here the gotTo animation has finished
    });
```

ArcGIS API for JavaScript

Working with the 3D SceneView

SceneView

The 3D View

- The `SceneView` provides 3D specific functionality

```
class SceneView {
  // Camera specifies the view
  camera: Camera;

  // Programmatic navigation
  goTo(...);

  // Settings that affect constraints (e.g. navigation constraints)
  constraints: SceneViewConstraints;

  // Padding on the view
  padding: { top: number, right: number, bottom: number, left: number };

  // Quality profile
  qualityProfile: string;

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

SceneView

Camera Definition

- 3D viewing parameters in a `SceneView` are controlled by `esri/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;
}
```

SceneView

Camera Interaction

- Changing `SceneView.camera` immediately updates the 3D view

```
// Get a copy of the current camera
var 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;
```

SceneView

View Navigation

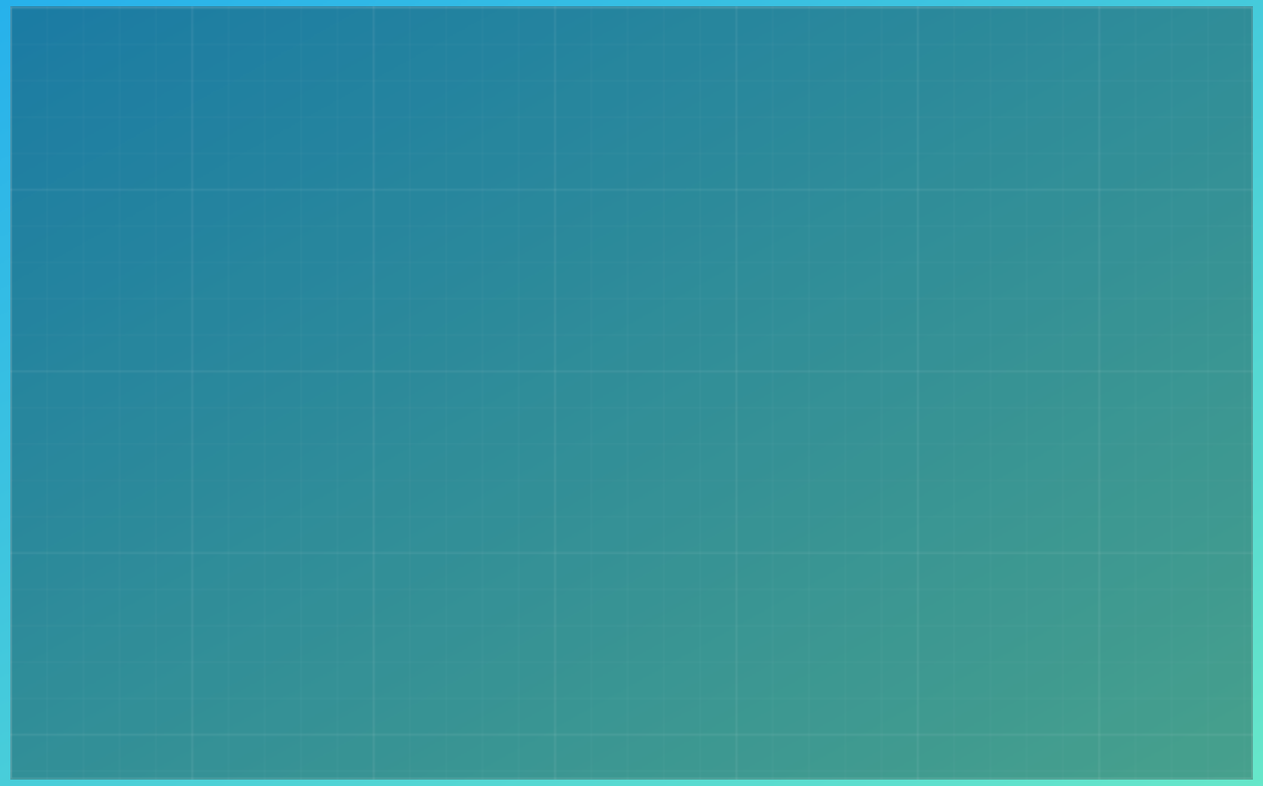
- Use `SceneView.goTo(target[, options])` to navigate
  - Supports different targets: `Camera`, `Geometry`, `Geometry[]`, `Graphic`, `Graphic[]`
  - Supports specifying desired `scale`, `position`, `heading` and `tilt`
  - Allows specifying animation options: `animate`, `speedFactor` or `duration`, `easing`
  - Returns a `Promise` which resolves when the animation has finished

SceneView

View Navigation #1

- Use `SceneView.goTo()` to view a set of graphics at a certain scale, heading and tilt

```
// Specify a target and additional
// parameters to further control the view
view.goTo({

  // The target is a set of graphics which should be
  // brought into view
  target: view.graphics

  // Additionally, define at which scale, heading and tilt
  // these graphics should be viewed
  scale: 5000,
  heading: 30,
  tilt: 60
});
```

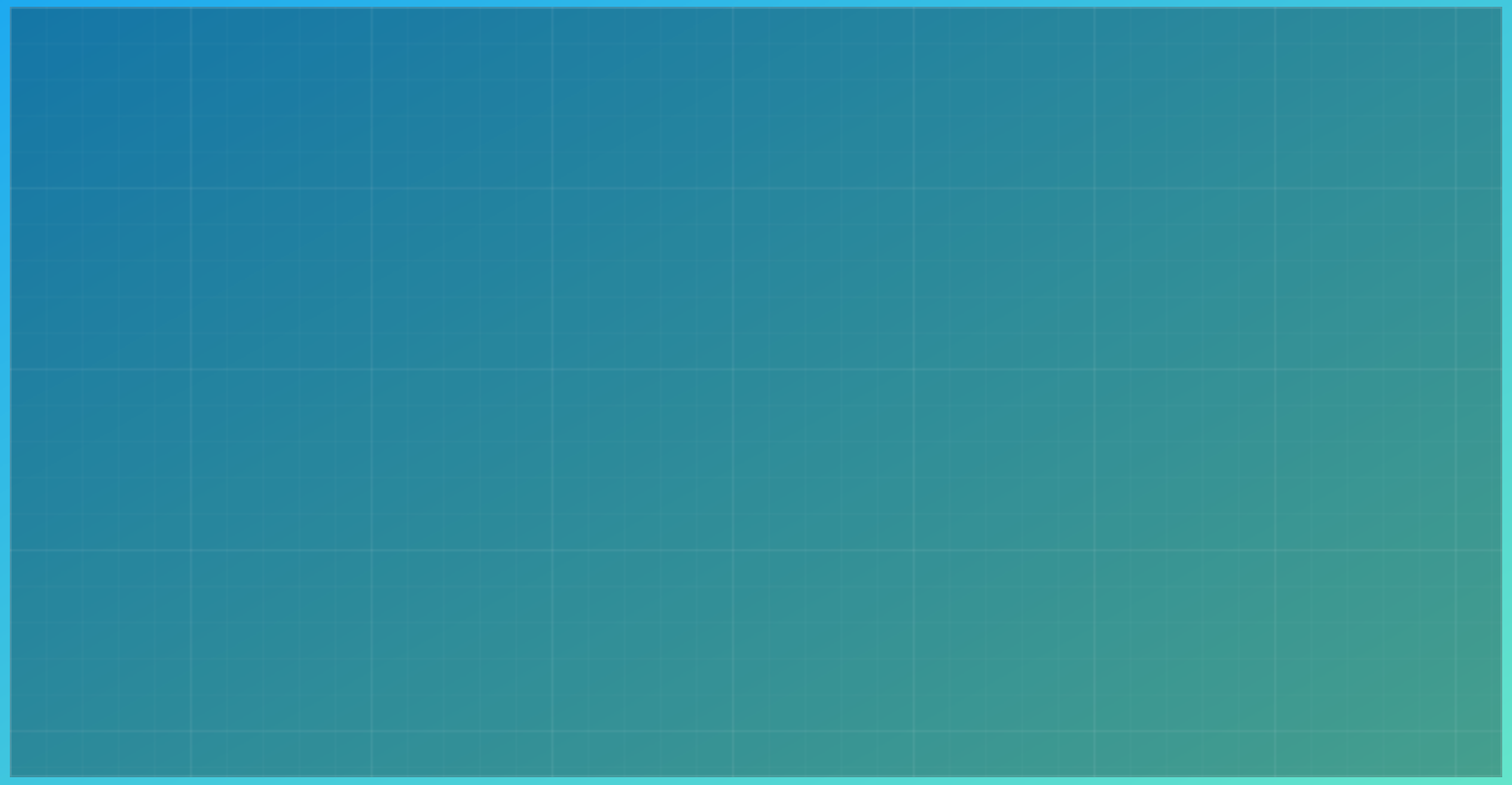


SceneView

View Navigation #2

- Use `SceneView.goTo()` to create a smooth camera animation on the 3D view

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

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

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



SceneView

View Navigation #3

- Use `SceneView.goTo()` `options` to control the animation

```
function rotateView(easing) {
  var cam = view.camera.clone();
  cam.position = {
      longitude: cam.position.longitude + 90,
      ...
  };

  return view.goTo({
    target: cam
  }, {
    easing: easing,
    ...
  });
}

rotateView("linear")

.then(function() {
  return rotateView("in-out-cubic");
})

.then(function() {
  rotateView("in-out-expo");
});
```

SceneView

View Constraints

- Use `SceneView.constraints` to control navigation and rendering aspects

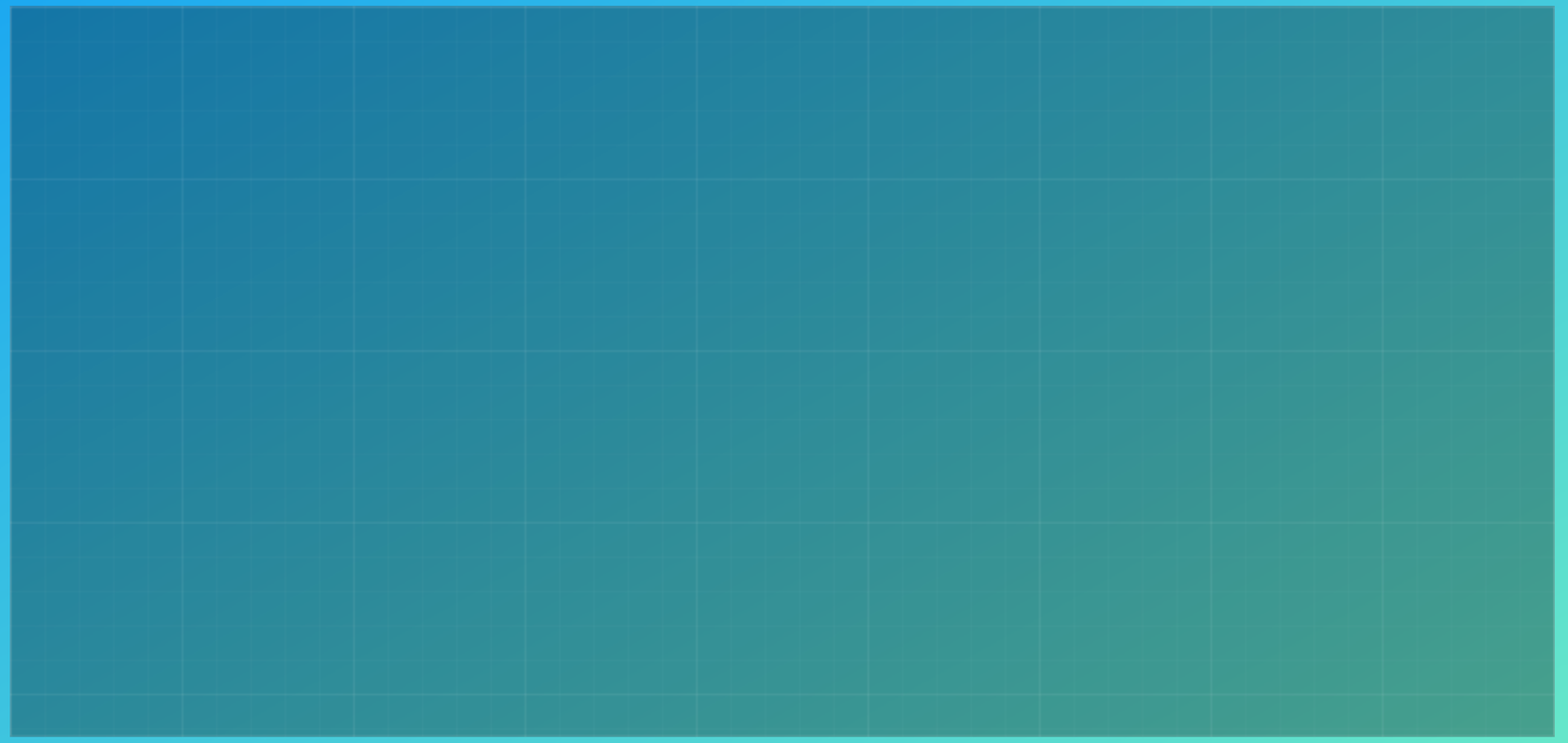
```
// The minimum and maximum allowed
// altitude (camera.position.z) of the camera.
view.constraints.altitude = {
  min: 10000000, // 10'000 km
  max: 30000000 // 30'000 km
};

// Set the clip distance near/far values
// to override the default clipping heuristics
view.constraints.clipDistance = {
  near: 10000000, // 10'000 km
  far: 40000000 // 40'000 km
};
```

SceneView

View Padding

- Use `SceneView.padding` to focus on a subsection of the view
- Primarily affects UI and navigation

```
// Set the padding to make
// space for a sidebar and a header
view.padding = {
  top: 50,
  left: 150
};
```



SceneView

View Quality

- Use `qualityProfile` and `quality` to control performance and quality
- Affects: Map resolution, scene level detail, anti-aliasing, atmosphere

```
viewLeft = new SceneView({
  qualityProfile: "low",

  environment: {
    atmosphere: { quality: "low" },
    lighting: {
      directShadowsEnabled: false,
      ambientOcclusionEnabled: false
    }
  }
});

viewRight = new SceneView({
  qualityProfile: "high"

  environment: {
    atmosphere: { quality: "high" }
    lighting: {
      directShadowsEnabled: true,
      ambientOcclusionEnabled: true
    }
  }
});
```

ArcGIS API for JavaScript

Adding Graphics to the 3D SceneView

Tectonic Plates & Boundaries

Dataset

The dataset presents tectonic plates and their boundaries, and in addition orogens and information about the boundaries. The data is useful for geological applications, analysis and education, and should be easy to use in any modern GIS software application.

Source: https://github.com/fraxen/tectonicplates

SceneView

Custom Graphics Layer

- Use `GraphicsLayer` to add manually created graphics to a 3D visualization

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

var graphicsLayer = new GraphicsLayer({
  elevationInfo: {
    mode: "on-the-ground"
  }
});

map.add(graphicsLayer);

var view = new SceneView({
  map: map,
  container: "viewDiv",

  center: [0, 0]
});
```

SceneView

Adding Graphics to the GraphicsLayer

```
view.when(function() {
  require(["dojo/text!./data/PB2002_plates.json"], function(PB2002_plates) {
    var plates = JSON.parse(PB2002_plates);

    for (var feature of plates.features) {

        var polygon = new Polygon({
          spatialReference: SpatialReference.WGS84,
          rings: feature.geometry.coordinates
        });

        var color = [ Math.round(Math.random() * 128) + 128,
                      Math.round(Math.random() * 128) + 128,
                      Math.round(Math.random() * 128) + 128, 0.5 ];

        var symbol = {
          type: "simple-fill",
          color: color,
          outline: { color: [255, 255, 255], width: 1 }
        };

        var graphic = new Graphic({
          geometry: polygon,
          symbol: symbol
        });

        graphicsLayer.add(graphic);
    }
  });
});
```

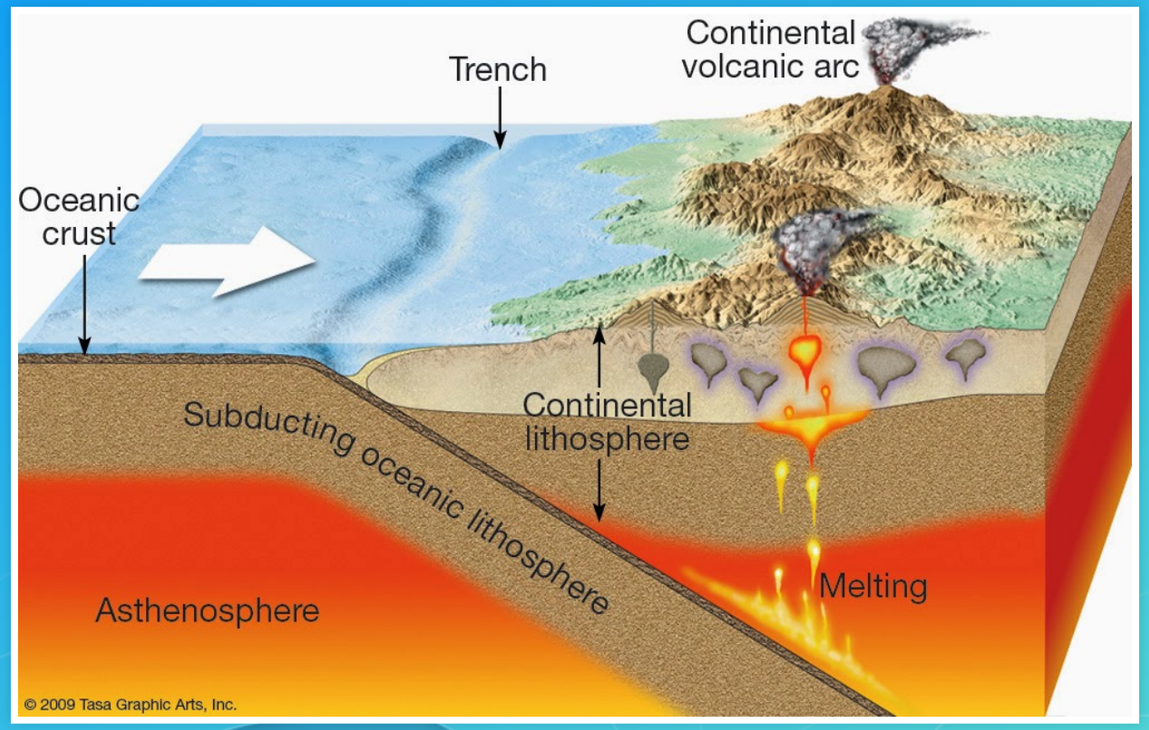
Building a customized tectonic plate visualization

Exploring the data

- Import GeoJSON into ArcGIS Online
- http://zurich.maps.arcgis.com/home/content.html

Visualization idea

- Tectonic plate boundary in context of the surface
- Tectonic plate subduction visualization



http://www.geologyin.com/2014/11/granites-and-convergence-zones-example.html

Step 1: Setting things up

- Local scene with clipping area
- Satellite imagery basemap
- World elevation
- Plates boundary line

Step 2: Geographic context

- Add small overview map

Step 3: Infographic style

- Add title and description using DOM
- Integrate it in the 3D view (2.5D)

Step 4: Boundary elevation profile

- Interested in seeing the elevation profile at plate boundary
- Integrated in the 3D scene as a wall (extruded line)
- **New**: Elevation sampling
- **New**: Mesh geometry

Elevation sampling

- Existing: `ElevationLayer.queryElevation`
- New in 4.7: View and offline elevation samplers
- Create an elevation cache for an extent from any elevation service (or the ground)
  - `ElevationLayer.createElevationSampler`
  - `view.groundView.elevationSampler`

```
const sampler = await elevationLayer.createElevationSampler(extent);

// After it has been created, sampling is synchronous
const z = sampler.elevationAt(point);
const polylineWithZ = sampler.queryElevation(polyline);
```

Mesh geometry

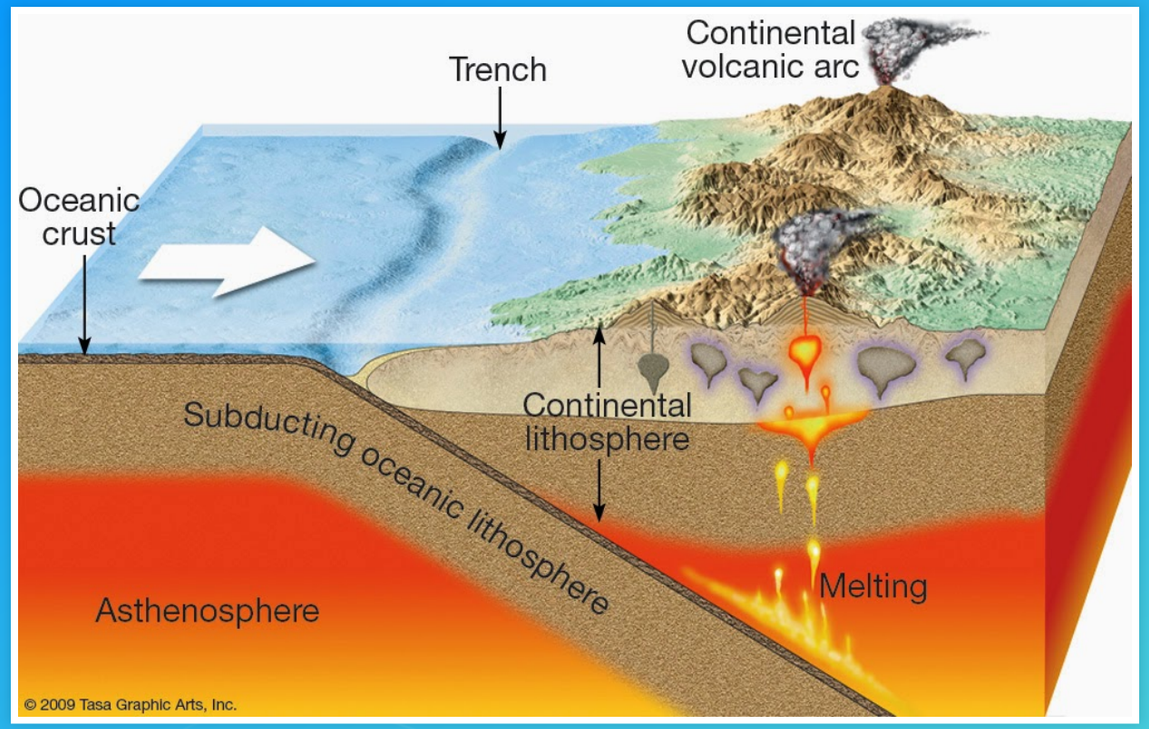
- A new client-side `esri/geometry` for 3D shapes
- General triangle soup representation
- Relatively low level
- Supports normals, textures and multiple components/materials
- Can be used with `MeshSymbol3D` and `FillSymbol3DLayer`

http://localhost/arcgis-js-sdk/build-artifacts/api-reference/esri-geometry-Mesh.html

Step 4: Boundary elevation profile

- Use ground view elevation sampling
- Construct 3D mesh geometry as a wall

Step 5: Tectonic plate visualization

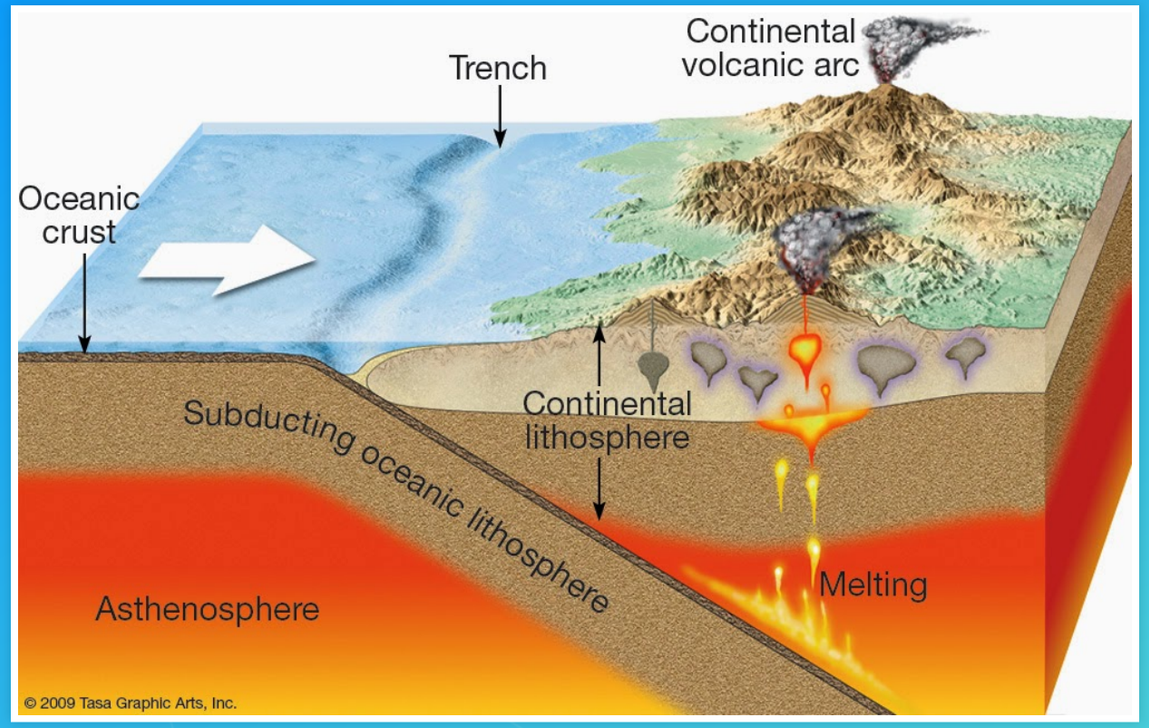


- Use velocity and orientation from the *steps* dataset
- Sample data at the edges of the clipping area
- Render a subsurface slice using `Mesh` geometry

Step 6: Slicing along the plate boundary

- We are only seeing a single location
- Interested in interactively slicing along the plate boundary
- Custom camera control along a path

Step 7: Finishing touches



- Lava style accentuates the plate subduction
- Would be interesting to have animated lava

External renderer

- You have data that you cannot visualize with available renderers, methods
- You want visualizations/animations that are not (yet) available
- You are familiar with WebGL and can afford the development effort
- **Important disclaimer: Experimental!**

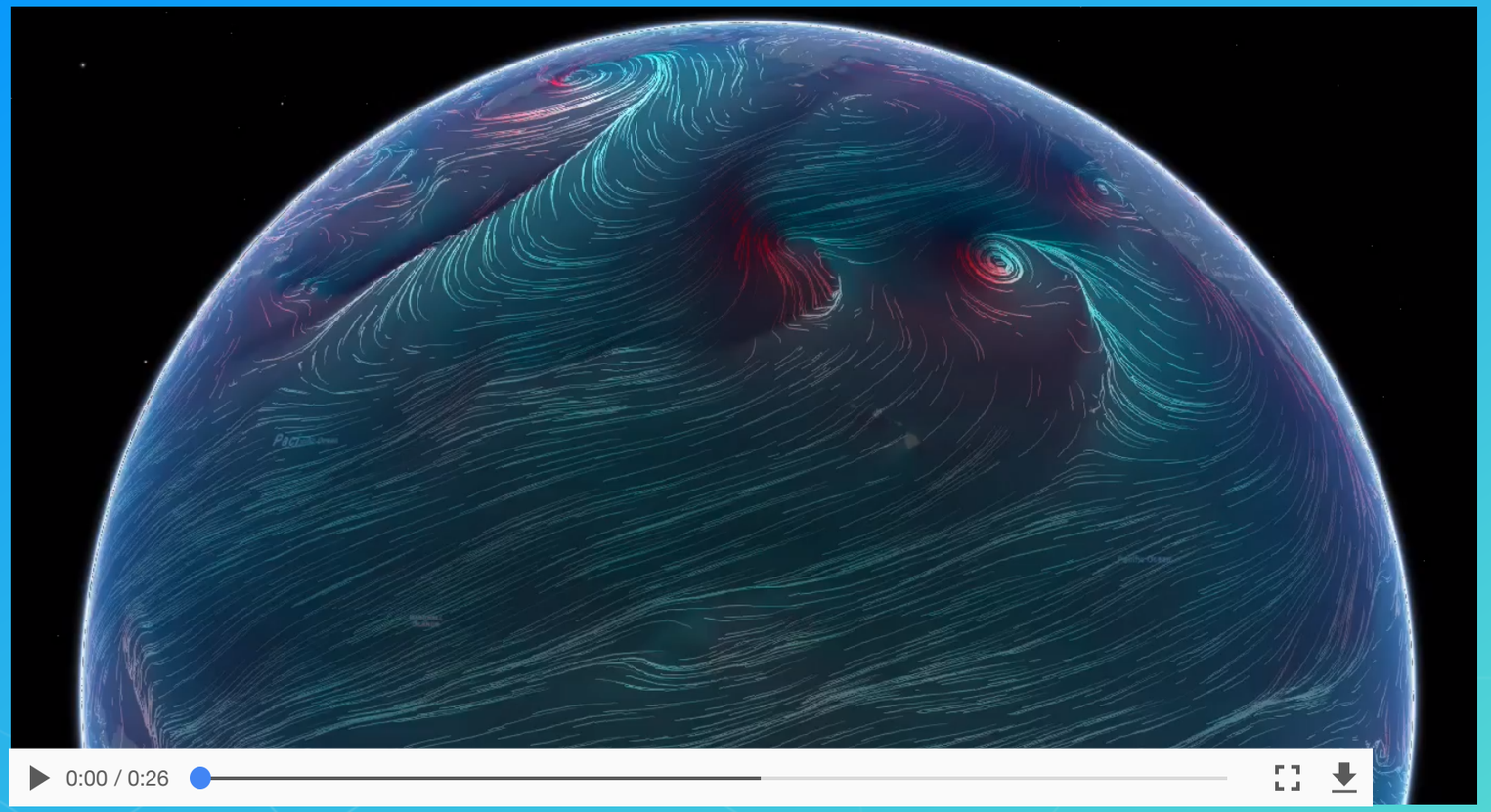
External renderer - examples

Windmills (JS API sample)

https://developers.arcgis.com/javascript/latest/sample-code/scene-external-renderer/index.html

External renderer - examples

Global wind currents



https://jkieboom.github.io/devsummit-palm-springs-2017/extending-3d-engine/#/28

External renderer: basic concepts and API

```
interface ExternalRenderer {
  setup(context: RenderContext): void;
  render(context: RenderContext): void;
}

interface RenderContext {
  gl: WebGLRenderingContext;
  camera: RenderCamera;

  resetWebGLState(): void;
}
```

https://developers.arcgis.com/javascript/latest/api-reference/esri-views-3d-externalRenderers.html

Step 7: Finishing touches

- Adding lava as an external renderer

esri

THE SCIENCE OF WHERE

https://github.com/jkieboom/devsummit-palm-springs-2018