cesiumpy Documentation

Release 0.0.1

sinhrks

October 17, 2016

Contents

Contents:

CHAPTER 1

Basics

This section describes the basic usage of `cesiumpy`. `cesiumpy` is the lightweight wrapper for Cesium.js.

This package offers limited API intended for:

- Interactive data visualization using `Jupyter Notebook`

This package IS NOT intended for:

- Website development using whole functionality of Cesium.js.

1.1 Installation

Use `pip`.

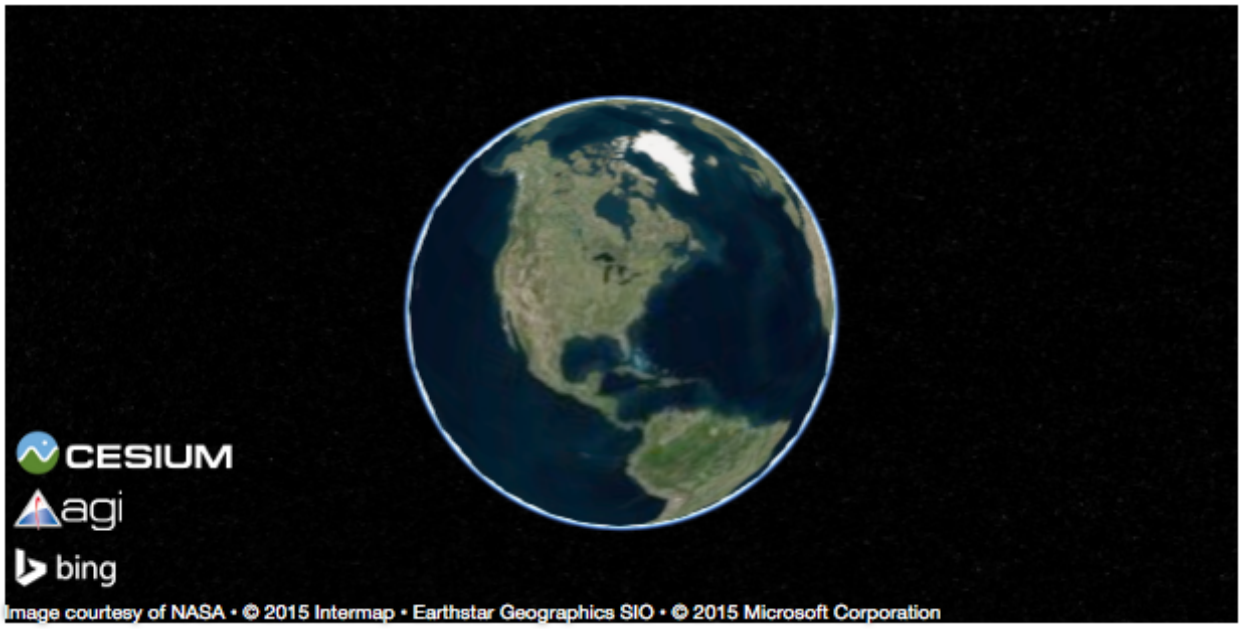
```
pip install cesiumpy
```

1.2 Display Cesium Widget

The easiest way to show the Cesium.js on your browser is to use `CesiumWidget` on `Jupyter Notebook`.

Because `CesiumWidget` has `_repr_html_` method to render `HTML` on `Jupyter Notebook`, placing variable contains `CesiumWidget` will output the map implemented on the output cell.

```
>>> import cesiumpy

>>> v = cesiumpy.CesiumWidget()
>>> v
```



If you do not use `Jupyter Notebook`, you can use `.to_html` method to output rendered `HTML`. Save the output as a file then open with your web browser.

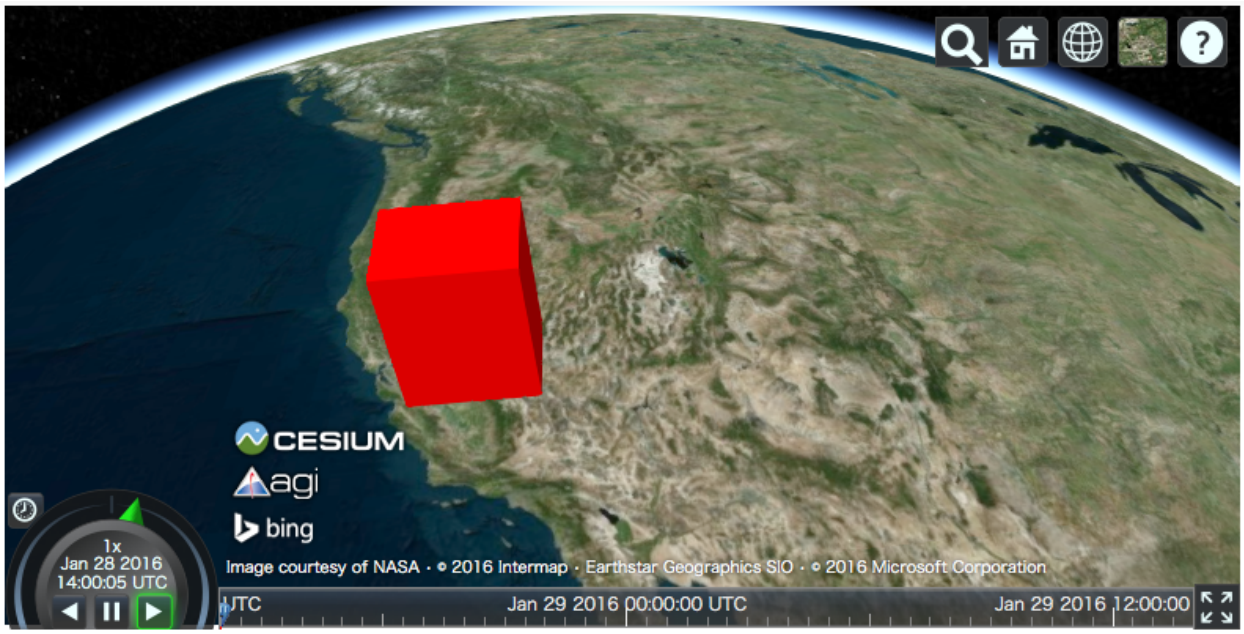
```
>>> v.to_html()
u'<script src="https://cesiumjs.org/Cesium/Build/Cesium/Cesium.js"></script>\n<link rel="stylesheet"
```

1.3 Add Entities

Cesium.js allows you to add various entities on the map. To do this, create `Viewer` instance and add preferable entity.

Even though `Viewer` also has various type of user control menus, below example disable almost of them because some of them are not displayed on `Jupyter Notebook` properly.

```
>>> v = cesiumpy.Viewer()
>>> b = cesiumpy.Box(dimensions=(40e4, 30e4, 50e4), material=cesiumpy.color.RED, position=[-120, 40,
>>> v.entities.add(b)
>>> v
```



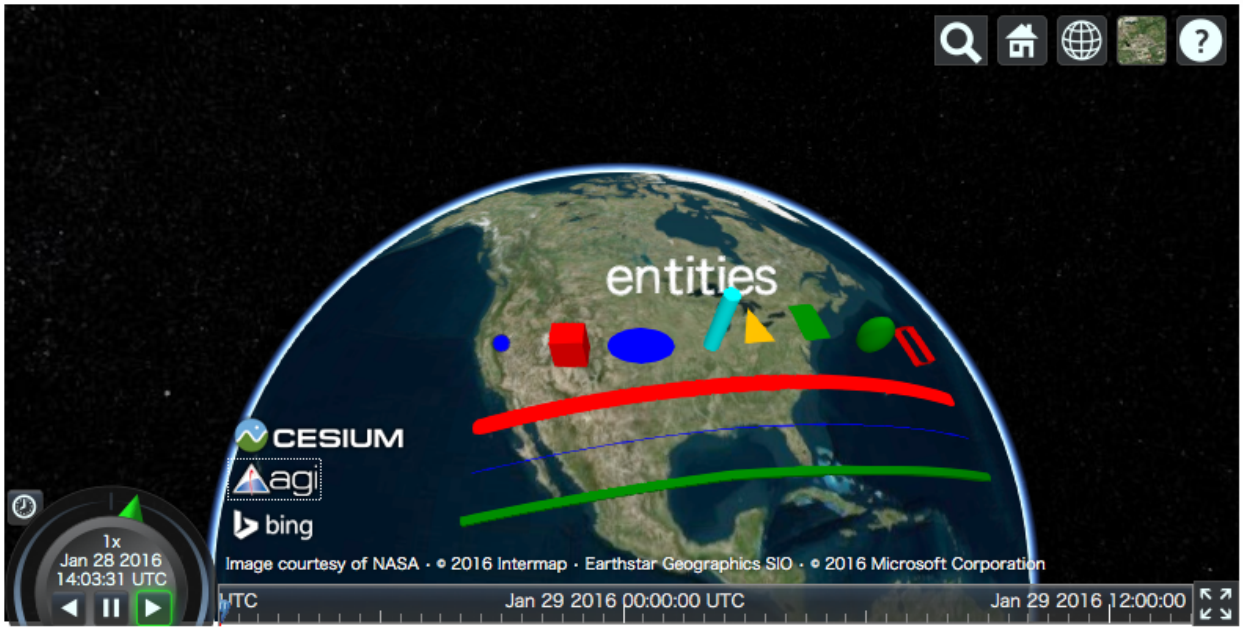
`cesiumpy` currently supports following entities. Refer to *Entities* section for more details.

- `Point`
- `Label`
- `Box`
- `Ellipse`
- `Cylinder`
- `Polygon`
- `Rectangle`
- `Ellipsoid`
- `Wall`
- `Corridor`
- `Polyline`
- `PolylineVolume`
- `Billboard`

The below example draws all entities on the map.

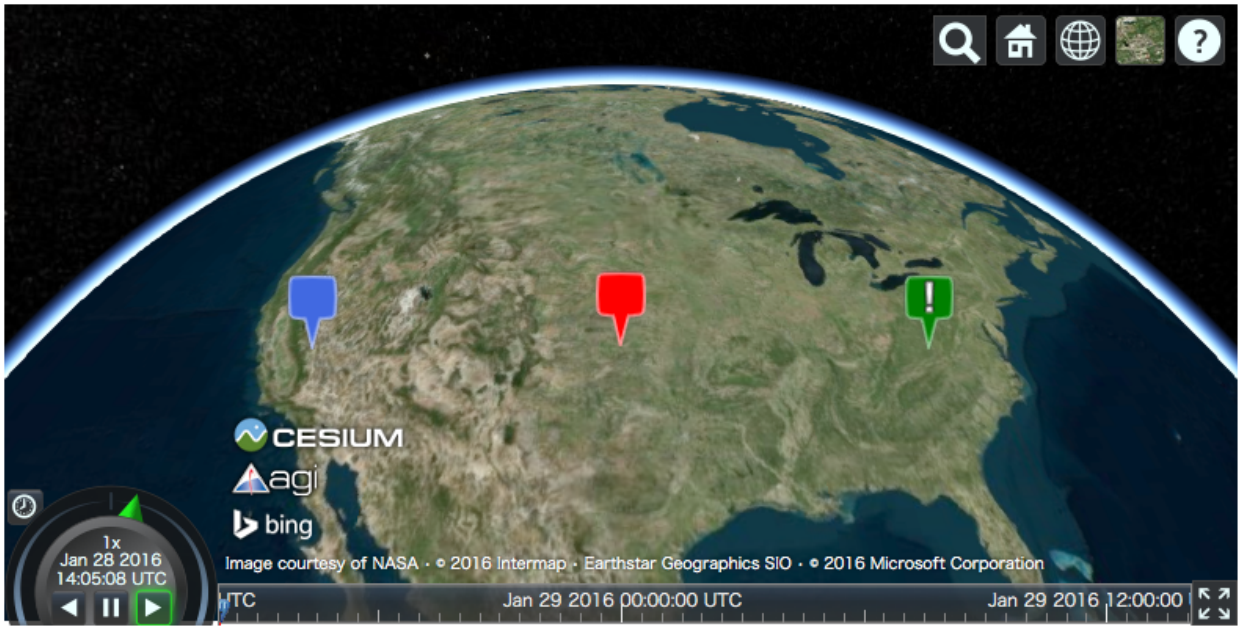
```
>>> v = cesiumpy.Viewer()

>>> label = cesiumpy.Label(position=[-90, 50, 0], text='entities')
>>> v.entities.add(label)

>>> point = cesiumpy.Point(position=[-120, 40, 0], color=cesiumpy.color.BLUE)
>>> v.entities.add(point)

>>> box = cesiumpy.Box(position=[-110, 40, 0], dimensions=(40e4, 30e4, 50e4), material=cesiumpy.colo
>>> v.entities.add(box)
```

```
>>> ellipse = cesiumpy.Ellipse(position=[-100, 40, 0], semiMinorAxis=25e4,
...                            semiMajorAxis=40e4, material=cesiumpy.color.BLUE)
>>> v.entities.add(ellipse)

>>> cylinder = cesiumpy.Cylinder(position=[-90, 40, 50e4], length=100e4,
...                              topRadius=10e4, bottomRadius=10e4,
...                              material=cesiumpy.color.AQUA)
>>> v.entities.add(cylinder)

>>> polygon = cesiumpy.Polygon(hierarchy=[-80, 40, -85, 40, -82.5, 45],
...                            material=cesiumpy.color.ORANGE)
>>> v.entities.add(polygon)

>>> rectangle = cesiumpy.Rectangle(coordinates=(-75, 40, -70, 45),
...                                material=cesiumpy.color.GREEN)
>>> v.entities.add(rectangle)

>>> ellipsoid = cesiumpy.Ellipsoid(position=(-60, 40, 0), radii=(20e4, 20e4, 30e4),
...                                material=cesiumpy.color.GREEN)
>>> v.entities.add(ellipsoid)

>>> wall = cesiumpy.Wall(positions=[-50, 35, -55, 35, -55, 40, -50, 40, -50, 35],
...                      maximumHeights=10e4, minimumHeights=0,
...                      material=cesiumpy.color.RED)
>>> v.entities.add(wall)

>>> corridor = cesiumpy.Corridor(positions=[-120, 30, -90, 35, -50, 30], width=2e5,
...                              material=cesiumpy.color.RED)
>>> v.entities.add(corridor)

>>> polyline = cesiumpy.Polyline(positions=[-120, 25, -90, 30, -50, 25], width=0.5, material=cesiumpy
>>> v.entities.add(polyline)

>>> polylinevolume = cesiumpy.PolylineVolume(positions=[-120, 20, -90, 25, -50, 20],
...                                          shape=[-5e4, -5e4, 5e4, -5e4, 5e4, 5e4, -5e4, 5e4],
...                                          material=cesiumpy.color.GREEN)
>>> v.entities.add(polylinevolume)
>>> v
```



The below example draws various `Pin` on the map.

```
>>> v = cesiumpy.Viewer()
>>> pin1 = cesiumpy.Pin()
>>> bill1 = cesiumpy.Billboard(position=[-120, 40, 0], image=pin1)
>>> v.entities.add(bill1)

>>> pin2 = cesiumpy.Pin(cesiumpy.color.RED)
>>> bill2 = cesiumpy.Billboard(position=[-100, 40, 0], image=pin2)
>>> v.entities.add(bill2)

>>> pin3 = cesiumpy.Pin.fromText('!!', color=cesiumpy.color.GREEN)
>>> bill3 = cesiumpy.Billboard(position=[-80, 40, 0], image=pin3)
>>> v.entities.add(bill3)
>>> v
```

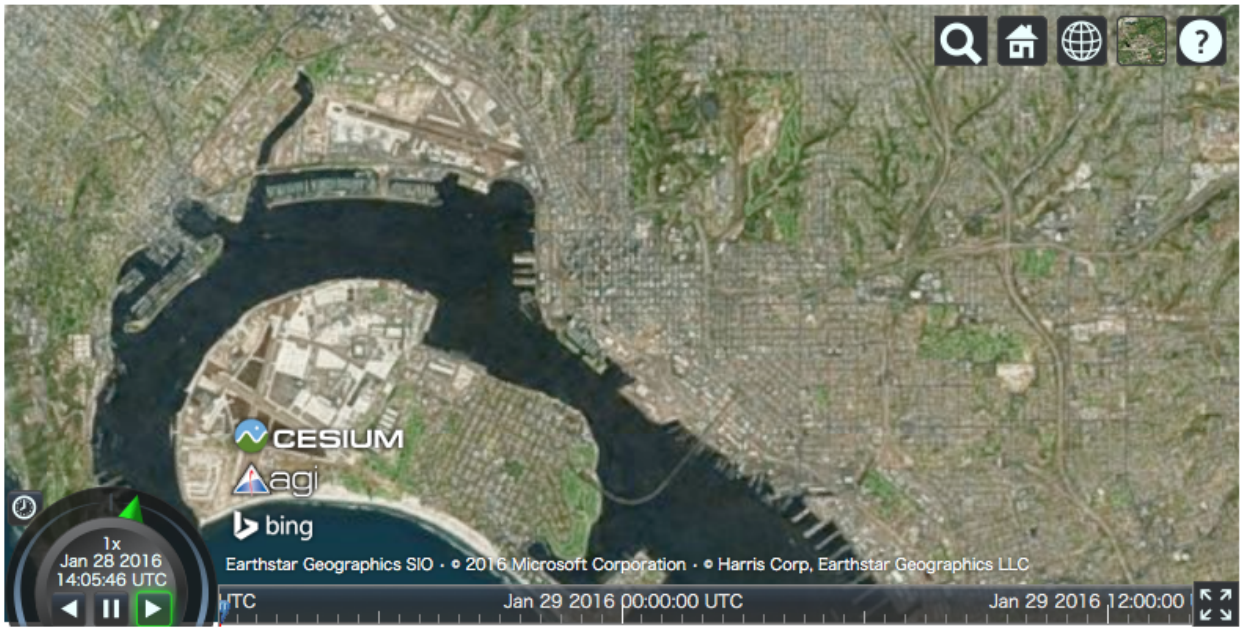


1.4 Camera

`CesiumWidget` and `Viewer` has a `camera` property which allows you to specify the location to be displayed. You can call `flyTo` method to specify the location passing `tuple` or `list`.

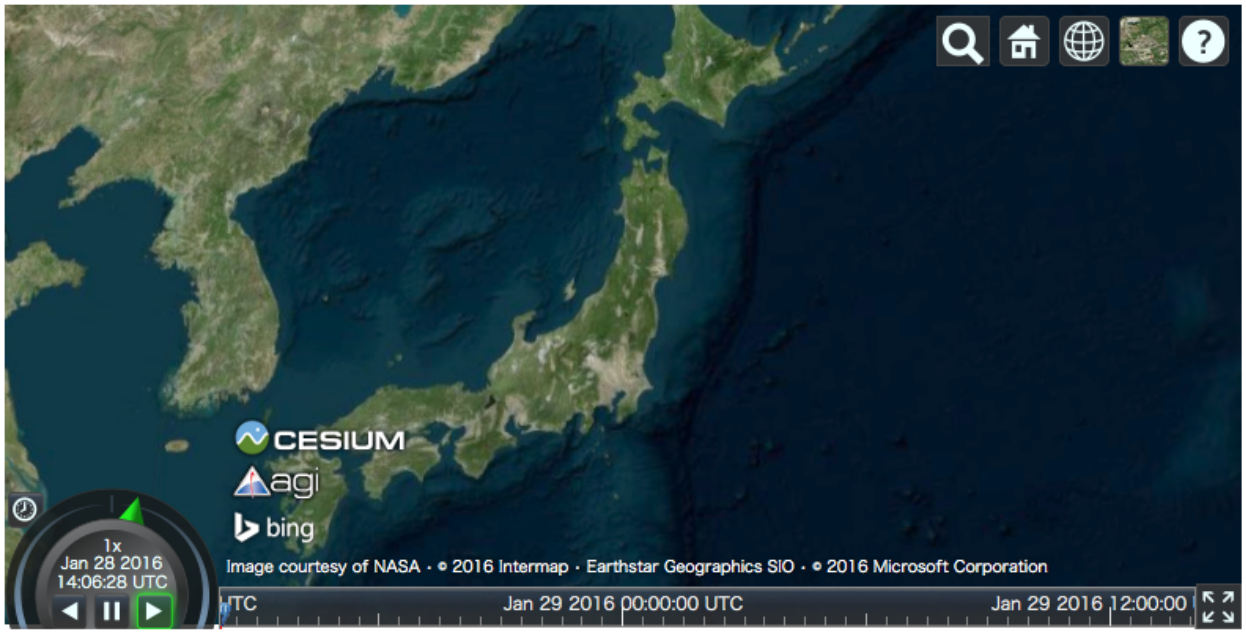
If input length is 3, it will be regarded as the point specified by (`longitude`, `latitude`, `height`).

```
>>> v = cesiumpy.Viewer()
>>> v.camera.flyTo((-117.16, 32.71, 15000))
>>> v
```



- If input length is 4, it will be regarded as the rectangle specified by (`west`, `south`, `east`, `north`).

```
>>> v = cesiumpy.Viewer()
>>> v.camera.flyTo((135, 30, 145, 45))
>>> v
```



1.5 Add Providers

Cesium.js supports some "layers" to cover the map. Objects which provides "layers" are called as "provider". There are 2 types of providers as below:

- `ImageryProvider`: Provides layers with imagery
- `TerrainProvider`: Provides layers with terrain and water effects

1.5.1 ImageryProvider

Refer to following document for the general explanation of `ImageryProvider`:

- http://cesiumjs.org/tutorials/Imagery-Layers-Tutorial/

The below example outputs the map covered by the image provided by the ArcGIS MapServer, as the same as the above tutorial.

```
>>> url = 'http://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer'
>>> imageryProvider = cesiumpy.ArcGisMapServerImageryProvider(url=url)

>>> v = cesiumpy.Viewer(imageryProvider=imageryProvider)
>>> v
```

Also you can use other providers.

```
>>> url = '//cesiumjs.org/tilesets/imagery/blackmarble'
>>> credit = 'Black Marble imagery courtesy NASA Earth Observatory'
>>> imageryProvider = cesiumpy.TileMapServiceImageryProvider(url=url, maximumLevel=8, credit=credit)

>>> v = cesiumpy.Viewer(imageryProvider=imageryProvider)
>>> v
```

Using `OpenStreetMapImageryProvider` can use OpenStreetMap as imagery.

```
>>> imageryProvider = cesiumpy.OpenStreetMapImageryProvider()
>>> v = cesiumpy.Viewer(imageryProvider=imageryProvider)
>>> v
```

1.5.2 TerrainProvider

Refer to following document for the general explanation of `TerrainProvider`:

- http://cesiumjs.org/tutorials/Terrain-Tutorial/

The below example outputs the map covered by the terrain provided by the Cesium Terrain Server, as the same as the above tutorial.

```
>>> url = '//assets.agi.com/stk-terrain/world'
>>> terrainProvider = cesiumpy.CesiumTerrainProvider(url=url)
>>> v = cesiumpy.Viewer(terrainProvider=terrainProvider)
>>> v
```

Passing `requestWaterMask=True` enables water effects.

```
>>> terrainProvider = cesiumpy.CesiumTerrainProvider(url=url, requestWaterMask=True)
>>> v = cesiumpy.Viewer(terrainProvider=terrainProvider)
>>> v
```

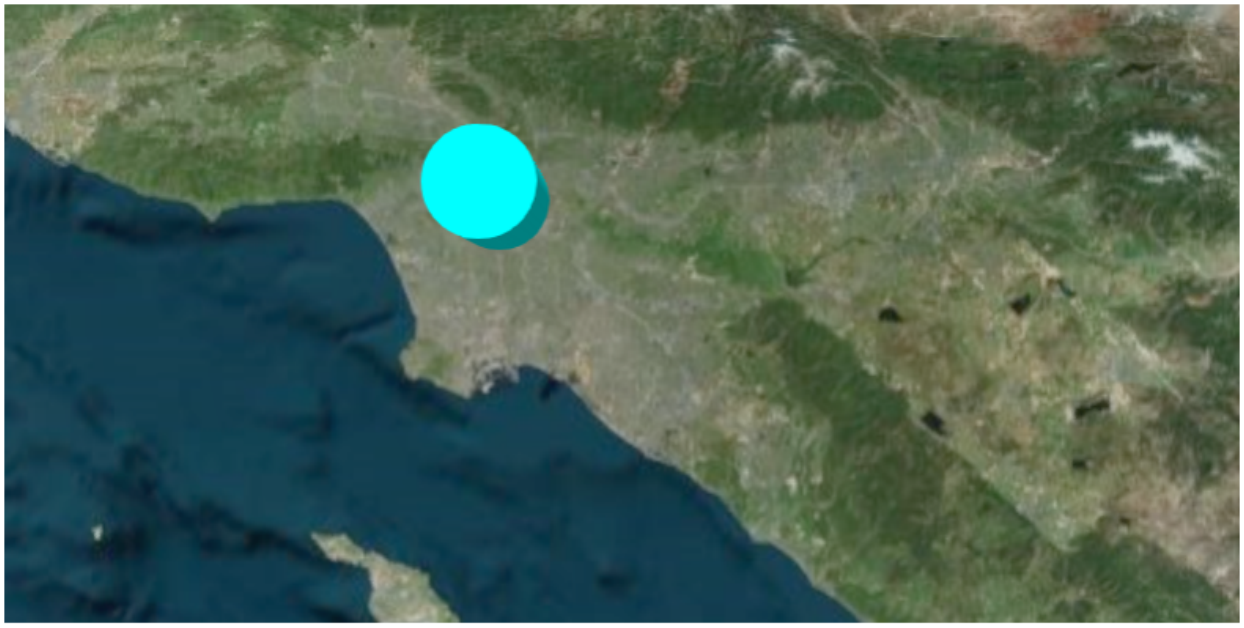
1.6 Geocoding

Cesium.js handles coordinates using numerics.

For convenience, `cesiumpy` automatically converts `str` input to coordinates via geocoding. The geocoding function is internally provided by `geopy`'s `GoogleV3` geocoder.

You can use `str` specifying location where you can use coordinates as below.

```
>>> viewer = cesiumpy.Viewer()
>>> cyl = cesiumpy.Cylinder(position='Los Angeles', length=30000, topRadius=10000,
...                         bottomRadius=10000, material='AQUA')
>>> v.entities.add(cyl)
>>> v.camera.flyTo('Los Angeles')
>>> v
```



CHAPTER 2

Entities

This section introduces various kinds of Entities supported by Cesium.js.

2.1 Cartesian

Cesium.js handles coordinates using `Cartesian` class. `Cartesian` may represent following 2 types of coordinates

- Pair of numerics, like `x`, `y`, `z`
- Geolocation (degrees), like `longitude`, `latitude`, `height`

```
>>> import cesiumpy

>>> cesiumpy.Cartesian2(10, 20)
Cartesian2(10, 20)

>>> cesiumpy.Cartesian3(10, 20, 30)
Cartesian3(10, 20, 30)

>>> cesiumpy.Cartesian3.fromDegrees(-110, 40, 0)
Cartesian3.fromDegrees(-110, 40, 0)
```

Basically you don't have to use the `Cartesian` classes because `cesiumpy` automatically converts python's `list` and `tuple` to `Cartesian` based on it's dimension.

2.2 Point

You can create the `Point` entity as below. `position` keyword can accepts 3 elements of `list` or `tuple` consists from longitude, latitude and height. `position` will be converted to `Cartesian3` automatically.

You can specify the color and size of the point via `color` and `pixelSize` keywords. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/PointGraphics.html

```
>>> p = cesiumpy.Point(position=[-110, 40, 0])
>>> p
Point(-110, 40, 0)

>>> p.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), point : {color : Cesium.Color.WHITE, pixelS

>>> p = cesiumpy.Point(position=[-110, 40, 0], color=cesiumpy.color.BLUE, pixelSize=20)
>>> p
Point(-110, 40, 0)

>>> p.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), point : {color : Cesium.Color.BLUE, pixelSi
```

The color constants are defined in `cesiumpy.color`, also you can specify it by name (`str`).

```
>>> p = cesiumpy.Point(position=[-110, 40, 0], color=cesiumpy.color.RED)
>>> p.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), point : {color : Cesium.Color.RED, pixelSiz

>>> p = cesiumpy.Point(position=[-110, 40, 0], color='blue')
>>> p.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), point : {color : Cesium.Color.BLUE, pixelSi
```

2.3 Label

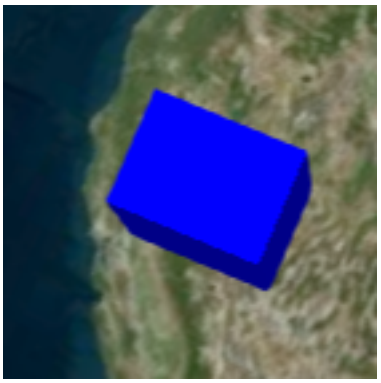
`Label` represents text displayed on the map. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/LabelGraphics.html

```
>>> l = cesiumpy.Label(position=[-110, 40, 0], text='xxx')
>>> l
Label(-110, 40, 0)

>>> l.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), label : {text : "xxx"}}'
```

```
>>> l = cesiumpy.Label(position=[-110, 40, 0], text='xxx', fillColor='red')
>>> l.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), label : {text : "xxx", fillColor : Cesium.C
```

2.4 Box



You can create the `Box` entity specifying its `position` and `dimensions` (size of each dimensions). Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/BoxGraphics.html

```
>>> b = cesiumpy.Box(position=[-110, 40, 0], dimensions=(40e4, 30e4, 50e4))
>>> b
Box(-110, 40, 0)

>>> b.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), box : {dimensions : new Cesium.Cartesian3(4

>>> b = cesiumpy.Box(position=[-110, 40, 0], dimensions=(10, 20, 30), material='blue')
>>> b.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), box : {dimensions : new Cesium.Cartesian3(1
```

2.5 Ellipse

Ellipse can be created by specifying its `position`, `semiMinorAxis` and `semiMajorAxis`. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/EllipseGraphics.html

```
>>> e = cesiumpy.Ellipse(position=[-110, 40, 0], semiMinorAxis=25e4,
...                      semiMajorAxis=40e4)
>>> e
Ellipse(-110, 40, 0)

>>> e.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), ellipse : {semiMinorAxis : 250000.0, semiMa

>>> e = cesiumpy.Ellipse(position=[-110, 40, 0], semiMinorAxis=100,
...                      semiMajorAxis=200, material='green')
>>> e.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), ellipse : {semiMinorAxis : 100, semiMajorAx
```

2.6 Cylinder

`Cylinder` can be created by its `position` and `length`. Note that its `position` must be specified with the center of the `Cylinder`. If you want to put the cylinder on the ground, `height` should be 100 if cylinder's `length` is 200. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/CylinderGraphics.html

```
>>> c = cesiumpy.Cylinder(position=[-110, 40, 100], length=200,
...                       topRadius=100, bottomRadius=100)
>>> c
Cylinder(-110, 40, 100)

>>> c.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 100), cylinder : {length : 200, topRadius : 100

>>> c = cesiumpy.Cylinder(position=[-110, 40, 250], length=500,
...                       topRadius=100, bottomRadius=100,
...                       material=cesiumpy.color.ORANGE)
>>> c.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 250), cylinder : {length : 500, topRadius : 100
```

2.7 Polygon

`Polygon` can be created by `positions` kw which specifies `list` of positions (`longitude`, `latitude` ...). The last position will be automatically connected to the first position. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/PolygonGraphics.html

```
>>> p = cesiumpy.Polygon(hierarchy=[-90, 40, -95, 40, -95, 45, -90, 40])
>>> p
Polygon([-90, 40, -95, 40, -95, 45, -90, 40])

>>> p.script
u'{polygon : {hierarchy : Cesium.Cartesian3.fromDegreesArray([-90, 40, -95, 40, -95, 45, -90, 40])}}
```

2.8 Rectangle

Rectangle can be created 4 elements of `list` or `tuple`, which represents south west longitude, south latitude, east longitude and north latitude. Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/RectangleGraphics.html

```
>>> r = cesiumpy.Rectangle(coordinates=(-85, 40, -80, 45))
>>> r
Rectangle(west=-85, south=40, east=-80, north=45)

>>> r.script
u'{rectangle : {coordinates : Cesium.Rectangle.fromDegrees(-85, 40, -80, 45)}}'
```

2.9 Ellipsoid

Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/EllipsoidGraphics.html

```
>>> e = cesiumpy.Ellipsoid(position=(-70, 40, 0), radii=(20e4, 20e4, 30e4))
>>> e
Ellipsoid(-70, 40, 0)

>>> e.script
u'{position : Cesium.Cartesian3.fromDegrees(-70, 40, 0), ellipsoid : {radii : new Cesium.Cartesian3(2
```

2.10 Wall

Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/WallGraphics.html

```
>>> w = cesiumpy.Wall(positions=[-60, 40, -65, 40, -65, 45, -60, 45],
...                   maximumHeights=10e4, minimumHeights=0)
>>> w
Wall([-60, 40, -65, 40, -65, 45, -60, 45])

>>> w.script
u'{wall : {positions : Cesium.Cartesian3.fromDegreesArray([-60, 40, -65, 40, -65, 45, -60, 45]), maxi
```

2.11 Corridor

Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/CorridorGraphics.html

```
>>> c = cesiumpy.Corridor(positions=[-120, 30, -90, 35, -60, 30], width=2e5)
>>> c
Corridor([-120, 30, -90, 35, -60, 30])

>>> c.script
u'{corridor : {positions : Cesium.Cartesian3.fromDegreesArray([-120, 30, -90, 35, -60, 30]), width :
```

2.12 Polyline

Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/PolylineGraphics.html

```
p = cesiumpy.Polyline(positions=[-120, 25, -90, 30, -60, 25], width=0.5)
>>> p
Polyline([-120, 25, -90, 30, -60, 25])

>>> p.script
u'{polyline : {positions : Cesium.Cartesian3.fromDegreesArray([-120, 25, -90, 30, -60, 25]), width :
```

2.13 PolylineVolume

Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/PolylineVolumeGraphics.html

```
>>> p = cesiumpy.PolylineVolume(positions=[-120, 20, -90, 25, -60, 20],
...                             shape=[-5e4, -5e4, 5e4, -5e4, 5e4, 5e4, -5e4, 5e4])
>>> p
PolylineVolume([-120, 20, -90, 25, -60, 20])

>>> p.script
u'{polylineVolume : {positions : Cesium.Cartesian3.fromDegreesArray([-120, 20, -90, 25, -60, 20]), sh
```

2.14 Billboard

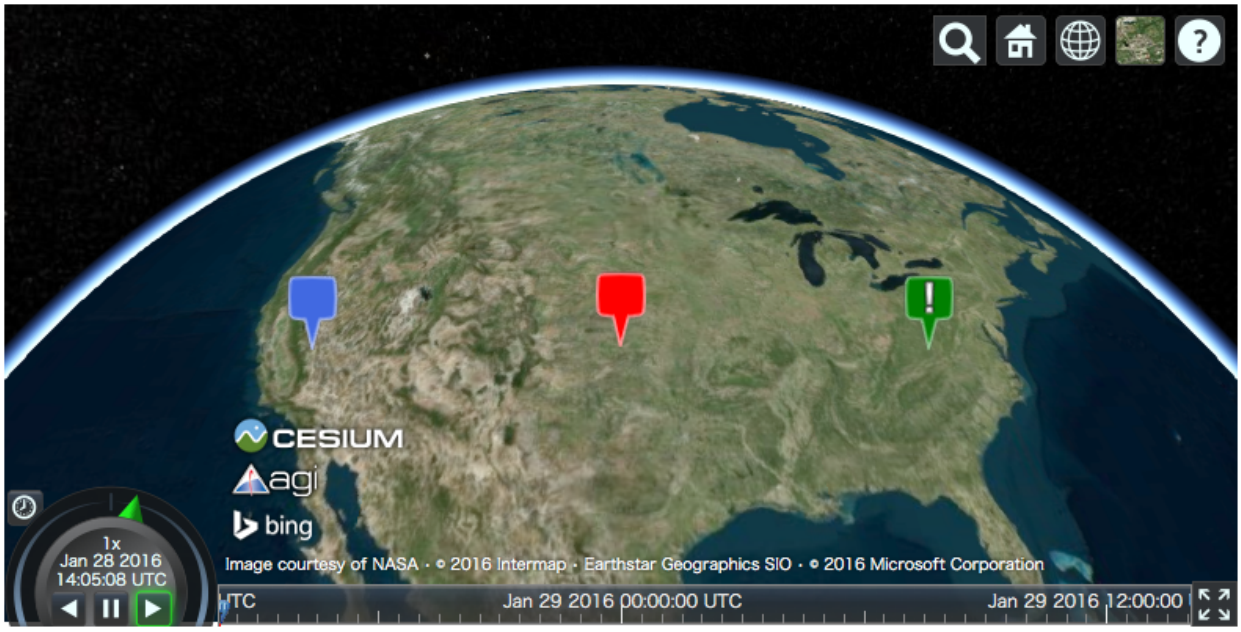
`Billboard` can dispaly a image on the map. Currently it supports to draw pins. You can pass `Pin` instance to the `Billboard` via `image` keyword.Refer to the following document to see the details of each options.

- https://cesiumjs.org/Cesium/Build/Documentation/BillboardGraphics.html

```
>>> p = cesiumpy.Pin()
>>> b = cesiumpy.Billboard(position=(-110, 40, 0), image=p)
>>> b
Billboard(-110, 40, 0)

>>> b.script
u'{position : Cesium.Cartesian3.fromDegrees(-110, 40, 0), billboard : {image : new Cesium.PinBuilder
```

You can change how `Pin` looks as below. Also, `Pin` can have label text to be displayed.

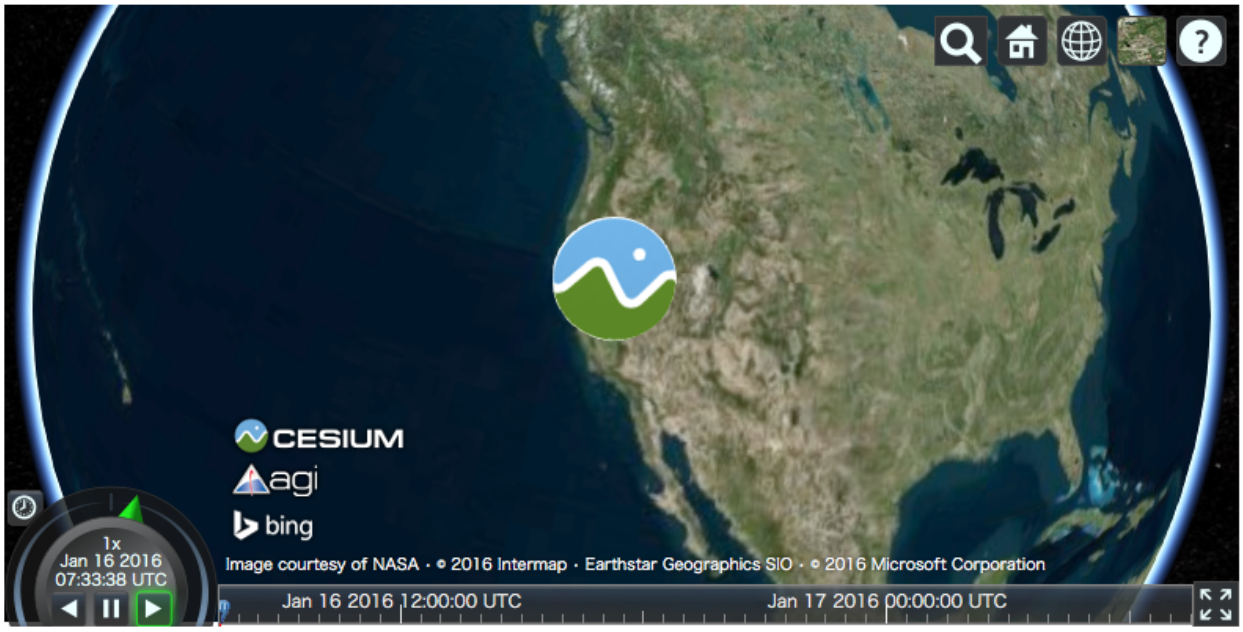
```
>>> v = cesiumpy.Viewer(**options)
>>> pin1 = cesiumpy.Pin()
>>> bill1 = cesiumpy.Billboard(position=[-120, 40, 0], image=pin1)
>>> v.entities.add(bill1)

>>> pin2 = cesiumpy.Pin(cesiumpy.color.RED)
>>> bill2 = cesiumpy.Billboard(position=[-100, 40, 0], image=pin2)
>>> v.entities.add(bill2)

>>> pin3 = cesiumpy.Pin.fromText('!', color=cesiumpy.color.GREEN)
>>> bill3 = cesiumpy.Billboard(position=[-80, 40, 0], image=pin3)
>>> v.entities.add(bill3)
>>> v
```



2.15 Material

You can use image file path via `material` keyword. The entity will be filled with the specified image.

```
v = cesiumpy.Viewer()
e = cesiumpy.Ellipse(position=(-120.0, 40.0, 0), semiMinorAxis=40e4,
                     semiMajorAxis=40e4, material='data/cesium_logo.png')
v.entities.add(e)
v
```


CESIUM
agi
bing
1x
Jan 16 2016
07:33:38 UTC
Image courtesy of NASA · © 2016 Intermap · Earthstar Geographics SIO · © 2016 Microsoft Corporation
Jan 16 2016 12:00:00 UTC
Jan 17 2016 00:00:00 UTC

CHAPTER 3

Plotting API

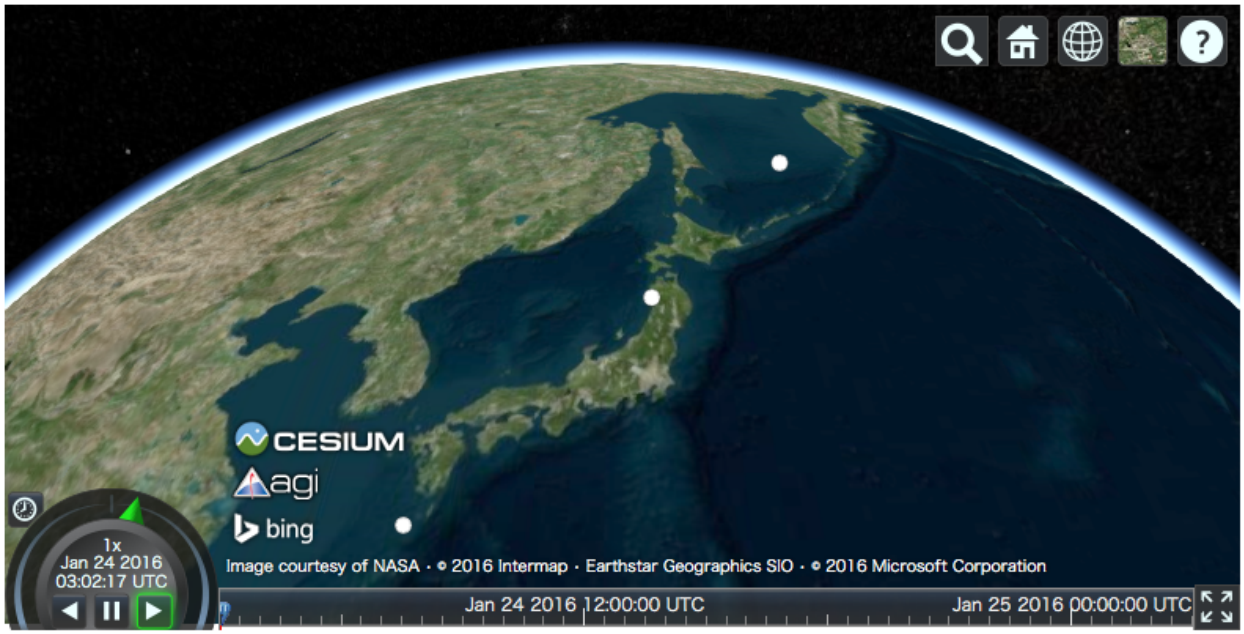
This section describes the plotting API of `cesiumpy`. Plotting API allows to add multiple entites at a time easily. You can call each plotting methods via `Viewer.plot` accessor.

3.1 Scatter

`Viewer.plot.scatter` draws multiple `Point` entity accepting following keywords. Both `x` and `y` must be provided at least.

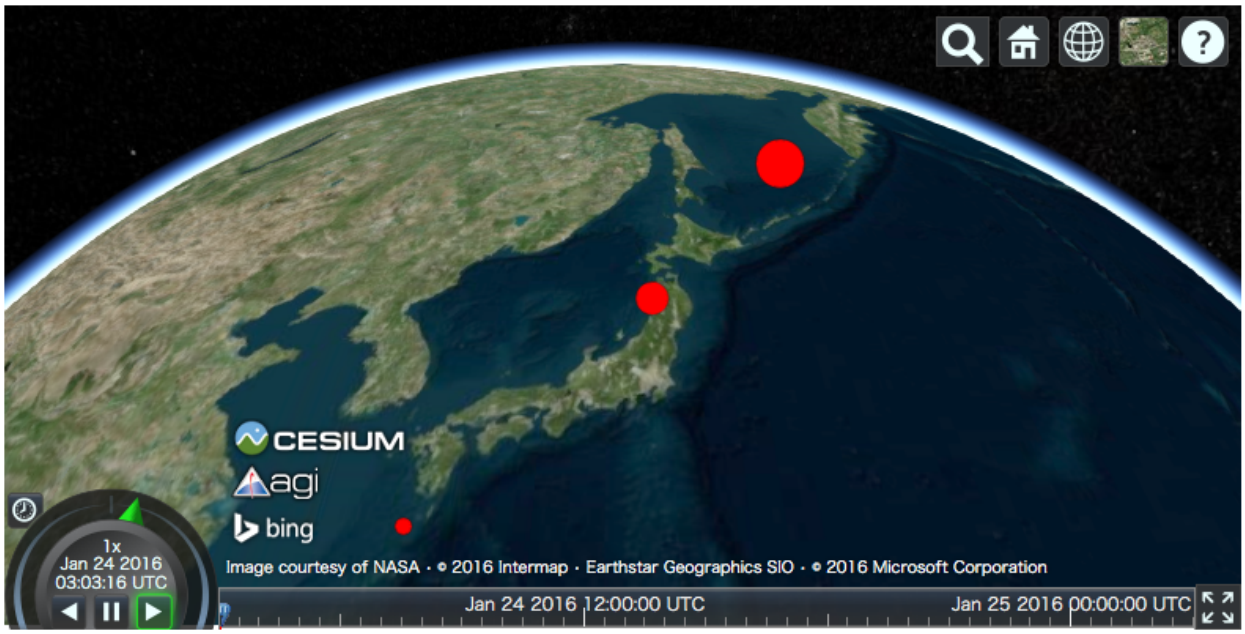
- `x`: Longitude
- `y`: Latitude
- `z`: Height
- `size`: Pixel size
- `color`: Color

```
>>> v = cesiumpy.Viewer()
>>> v.plot.scatter([130, 140, 150], [30, 40, 50])
```



For other keywords, you can use `list` or scalar. When `list` is passed, each element is used in the corresponding entity. If scalar is passed, all entities use the specified value.

```
>>> v = cesiumpy.Viewer()
>>> v.plot.scatter([130, 140, 150], [30, 40, 50],
...                size=[10, 20, 30], color=cesiumpy.color.RED)
```

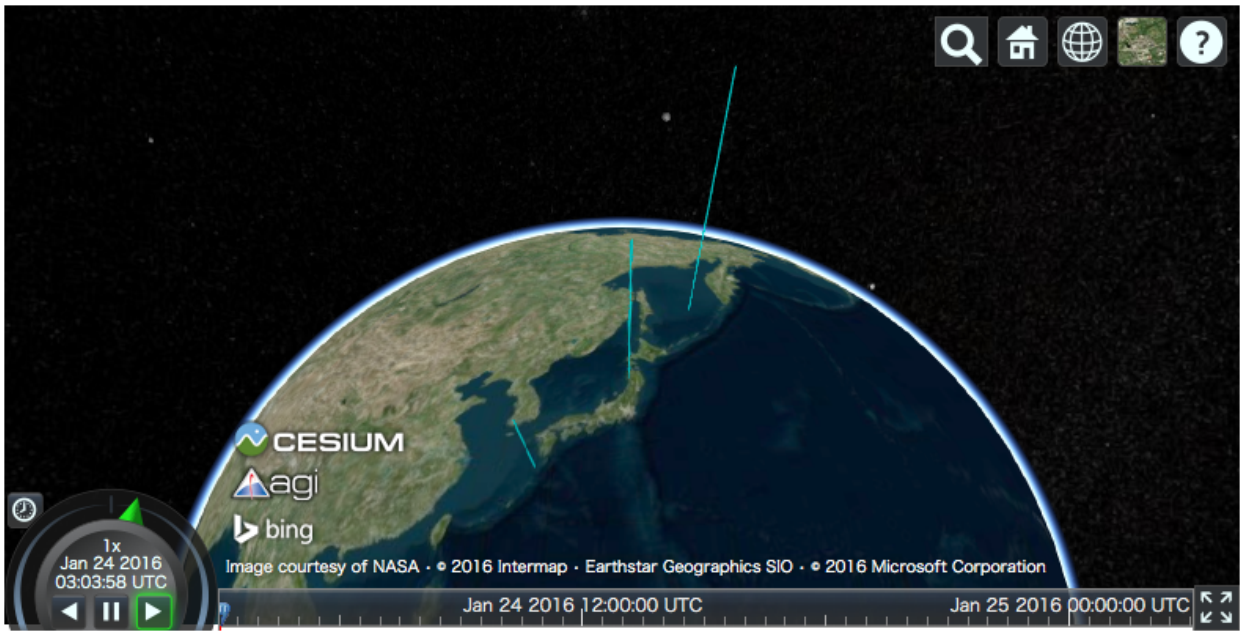


3.2 Bar

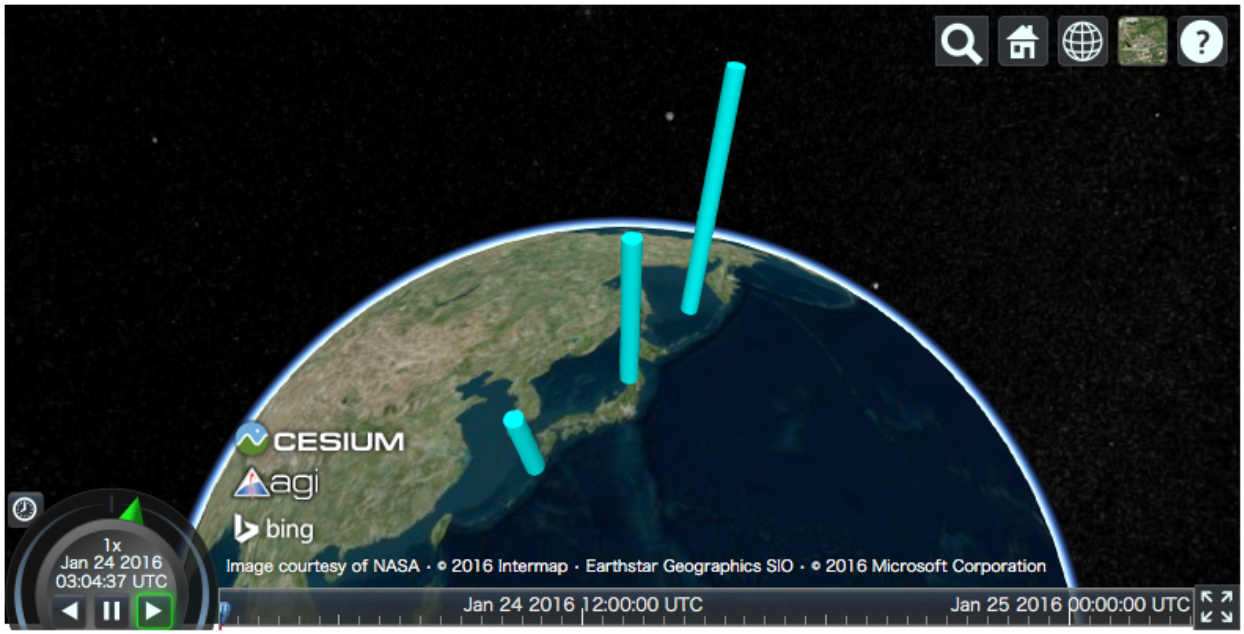
`Viewer.plot.bar` draws 3d bar using `Cylinder` entity accepting following keywords. `x`, `y` and `z` must be provided at least.

- `x`: Longitude
- `y`: Latitude
- `z`: Height
- `size`: Radius
- `color`: Color
- `bottom`: Bottom heights

```
>>> v = cesiumpy.Viewer()
>>> v.plot.bar([130, 140, 150], [30, 40, 50],
...            z=[10e5, 20e5, 30e5], color=cesiumpy.color.AQUA)
```

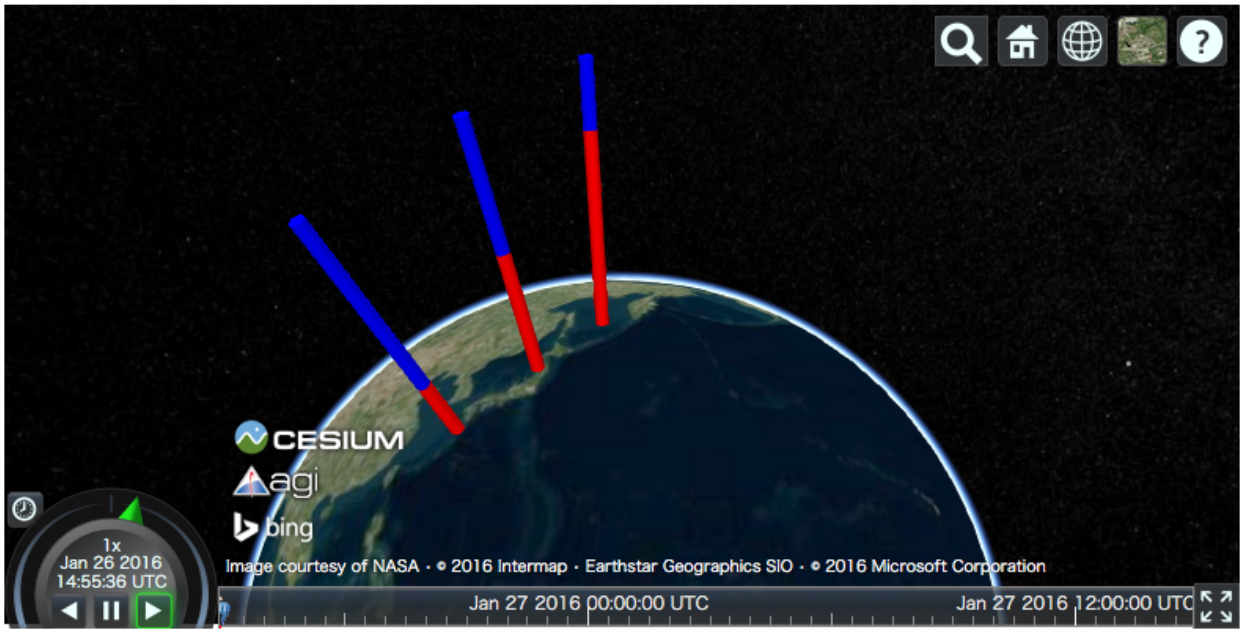



```
>>> v = cesiumpy.Viewer()
>>> v.plot.bar([130, 140, 150], [30, 40, 50], z=[10e5, 20e5, 30e5],
...            color=cesiumpy.color.AQUA, size=1e5)
```



Specifying `bottom` keyword allows to draw stacked bar plot.

```
>>> v = cesiumpy.Viewer('viewertest')
>>> v.plot.bar([130, 140, 150], [30, 40, 50], [1e6, 2e6, 3e6],
...            size=1e5, color=cesiumpy.color.RED)
>>> v.plot.bar([130, 140, 150], [30, 40, 50], [3e6, 2e6, 1e6],
...            size=1e5, color=cesiumpy.color.BLUE,
...            bottom=[1e6, 2e6, 3e6])
```

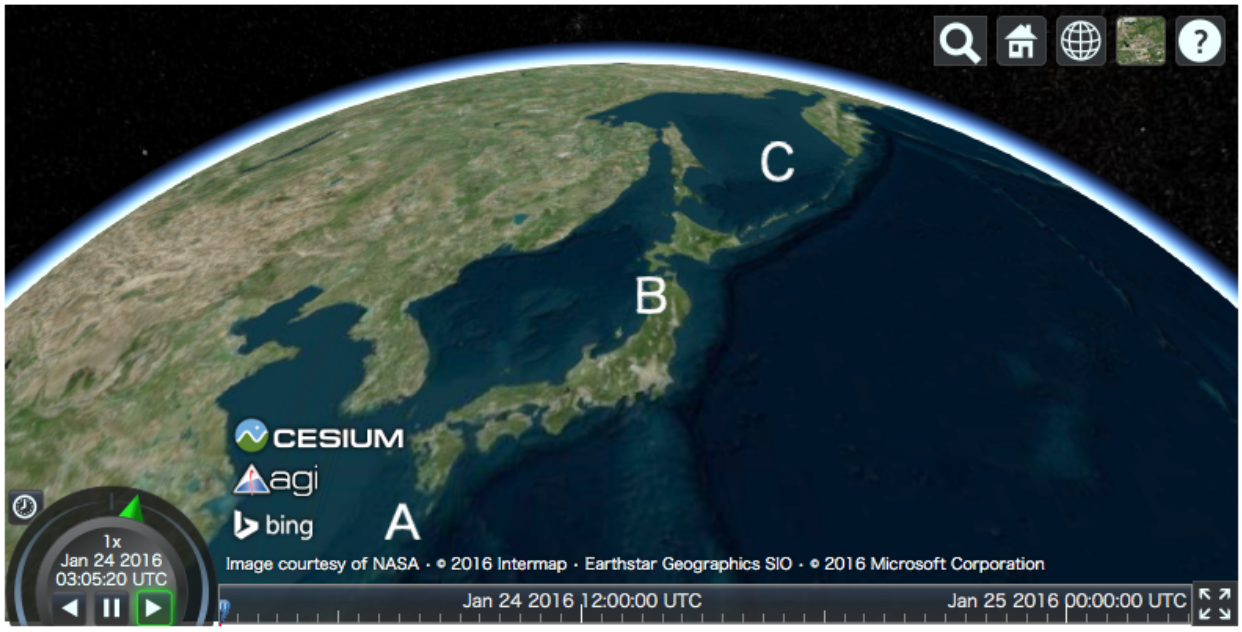


3.3 Label

`Viewer.plot.label` draws texts using `Label` entity accepting following keywords. `text`, `x` and `y` must be provided at least.

- `text`: Labels
- `x`: Longitude
- `y`: Latitude
- `z`: Height
- `size`: Text size
- `color`: Color

```
>>> v = cesiumpy.Viewer()
>>> v.plot.label(['A', 'B', 'C'], [130, 140, 150], [30, 40, 50])
```

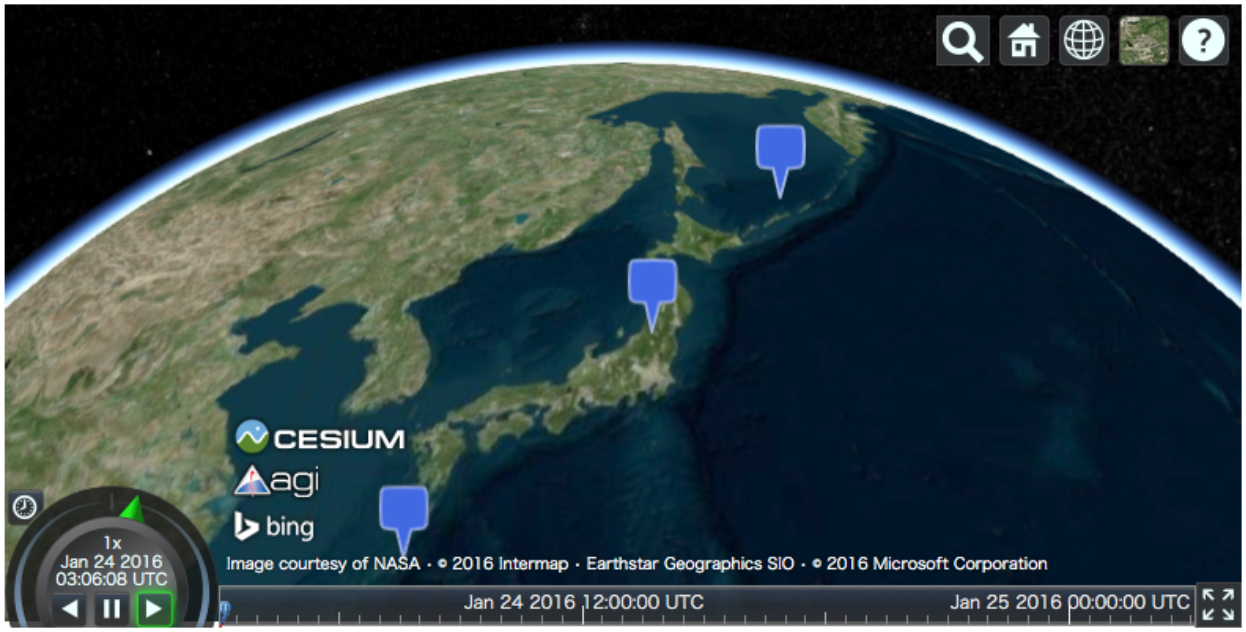


3.4 Pin

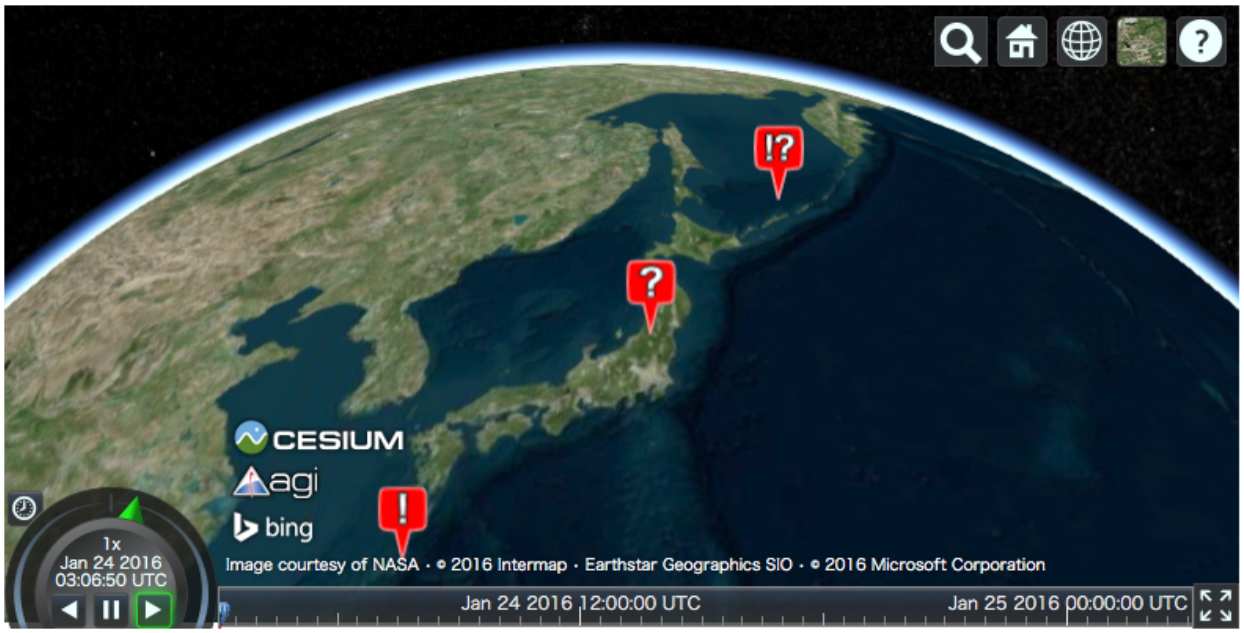
`Viewer.plot.pin` draws pins using `Billboard` entity accepting following keywords. Both `x` and `y` must be provided at least.

- `x`: Longitude
- `y`: Latitude
- `z`: Height
- `text`: Label
- `size`: Pin size
- `color`: Color

```
>>> v = cesiumpy.Viewer()
>>> v.plot.pin([130, 140, 150], [30, 40, 50])
```




```
>>> v = cesiumpy.Viewer()
>>> v.plot.pin([130, 140, 150], [30, 40, 50],
...            color=cesiumpy.color.RED, text=['!', '?', '!?'])
```



3.5 Specifying Color

You can colorize each entity using `cesiumpy.color.Color` instance. Common colors are defined under `cesiumpy.color`. Refer to Cesium Documentation to see the list of constants.

```
>>> cesiumpy.color.AQUA
Color.AQUA
```

Also, you can pass RGB or RGBA values to instanciate `Color`.

```
# RGB
>>> cesiumpy.color.Color(1, 0, 0)
Color(1.0, 0.0, 0.0)

# RGBA
>>> cesiumpy.color.Color(1, 0, 0, 0.5)
Color(1.0, 0.0, 0.0, 0.5)
```

If you want to use `str` representation, use `fromString` method.

```
>>> cesiumpy.color.Color.fromString("#FF0000")
Color.fromCssColorString("#FF0000")
```

There are 2 functions to prepare color at random.

- `choice`: Get a single color constant randomly.
- `sample`: Get a `list` of random color constants with specified length.

```
>>> cesiumpy.color.choice()
Color.DARKSLATEGREY

>>> cesiumpy.color.sample(3)
[Color.THISTLE, Color.PINK, Color.DARKKHAKI]
```

3.6 ColorMap

Also, `cesiumpy` has `ColorMap` class which internally uses `matplotlib ColorMap`. This is convenient to prepare multiple colors based on external values.

```
>>> cmap = cesiumpy.color.get_cmap('winter')
>>> cmap
ColorMap("winter")

>>> cmap(0.5)
Color(0.0, 0.501960784314, 0.749019607843, 1.0)

>>> cmap([0.2, 0.6])
[Color(0.0, 0.2, 0.9, 1.0), Color(0.0, 0.6, 0.7, 1.0)]
```

CHAPTER 4

Read External Files

4.1 Read External Files as Data Source

Cesium.js has a `DataSource` class which can draw external data as entities.

`cesiumpy` currently supports following `DataSource`.

- `GeoJsonDataSource`
- `KmlDataSource`
- `CzmlDataSource`

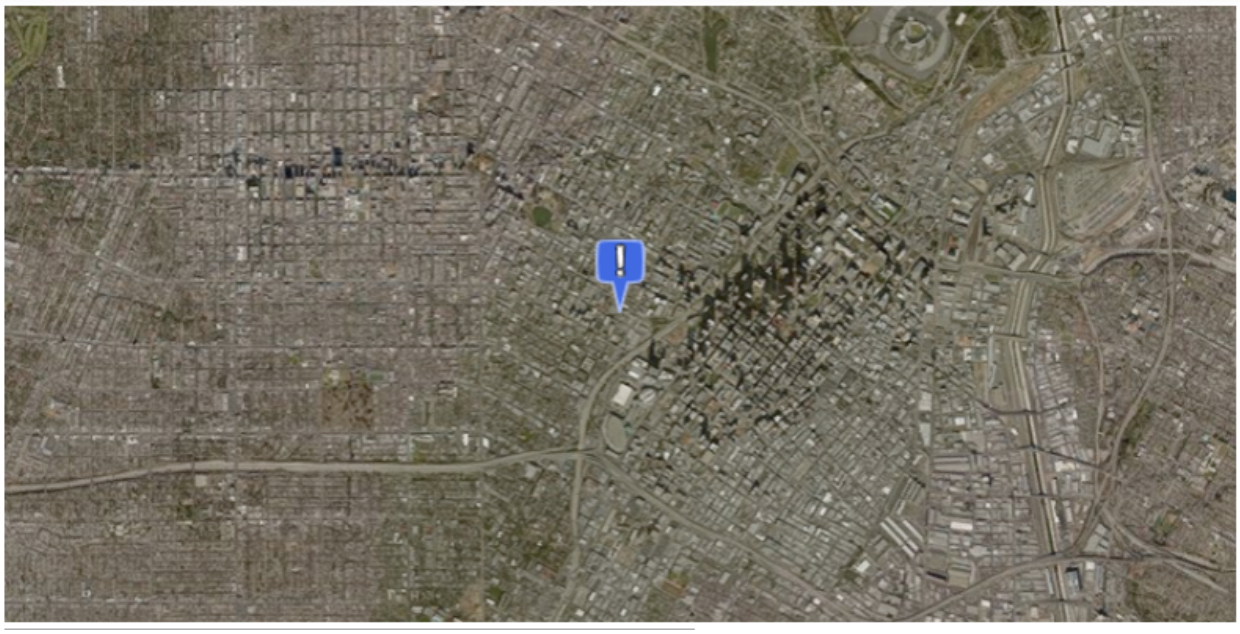
4.1.1 GeoJSON

Assuming we hanve following `.geojson` file named "example.geojson".

```
{
    "type": "Point",
    "coordinates": [-118.27, 34.05 ]
}
```

You can create `GeoJsonDataSource` instannce then add to `Viewer.DataSources`. `markerSymbol` option specifies the symbol displayed on the marker.

```
>>> ds = cesiumpy.GeoJsonDataSource('./example.geojson', markerSymbol='!')
>>> v = cesiumpy.Viewer()
>>> v.dataSources.add(ds)
>>> v
```



Or, you can use `load` class method to instanciate `DataSource` like `Cesium.js`.

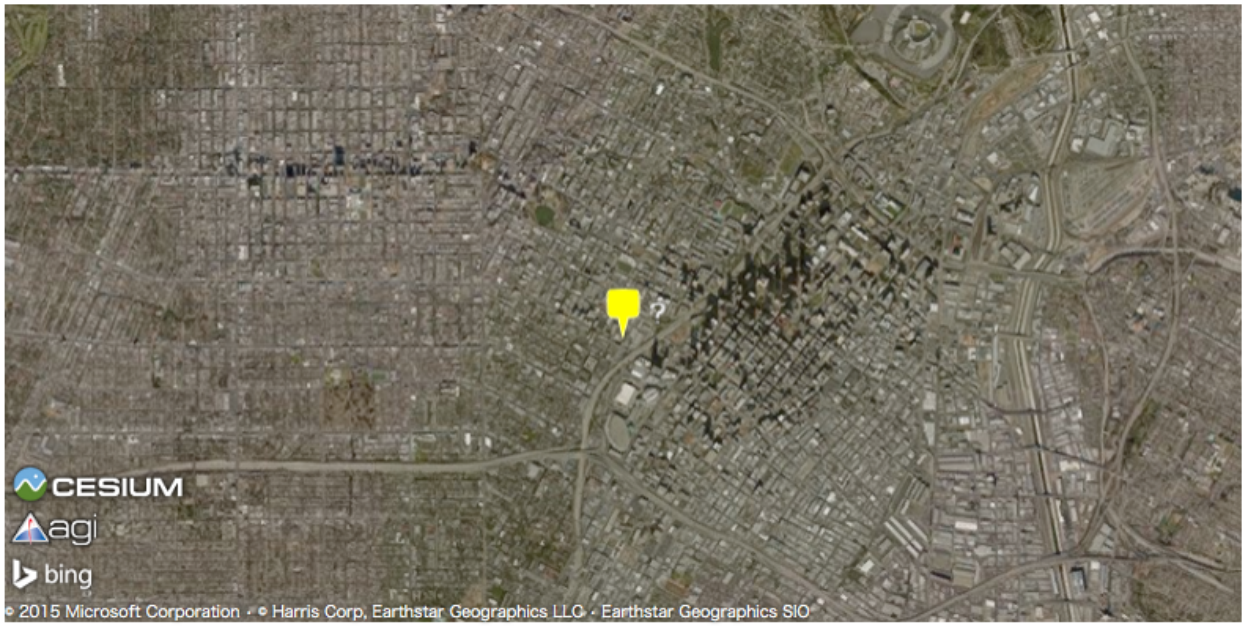
```
>>> cesiumpy.GeoJsonDataSource.load('./example.geojson', markerSymbol='!')
```

4.1.2 KML

You can use `KmlDataSource` to read `.kml` files. Assuming we have following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"> <Placemark>
 <name>?</name>
 <Point>
 <coordinates>-118.27,34.05,0</coordinates>
 </Point>
 </Placemark> </kml>
```

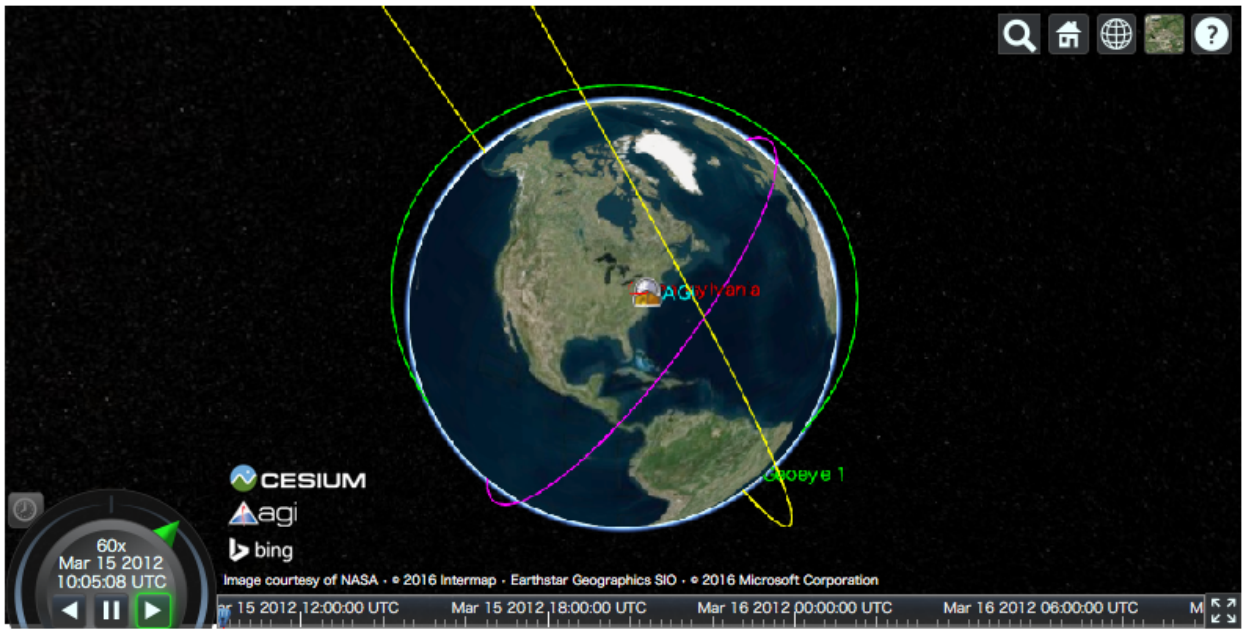
```
>>> ds = cesiumpy.KmlDataSource('example.kml')
>>> v = cesiumpy.Viewer()
>>> v.dataSources.add(ds)
>>> v
```



4.1.3 CZML

The last example is use `.czml` file downloaded from the Cesium.js repository

```
ds = cesiumpy.CzmlDataSource('sample.czml')
v = cesiumpy.Viewer()
v.dataSources.add(ds)
v
```



4.2 Read External Files as Entities

`cesiumpy` can read following file formats using `io` module. The results are automatically converted to `cesiumpy` entities and can be added to map directly.

- GeoJSON
- Shapefile

4.2.1 GeoJSON

This example reads GeoJSON file of Japanese land area. `cesiumpy.io.read_geojson` returns a `list` of `cesiumpy.Polygon`.

The file is provided by mledoze/countries repositry.

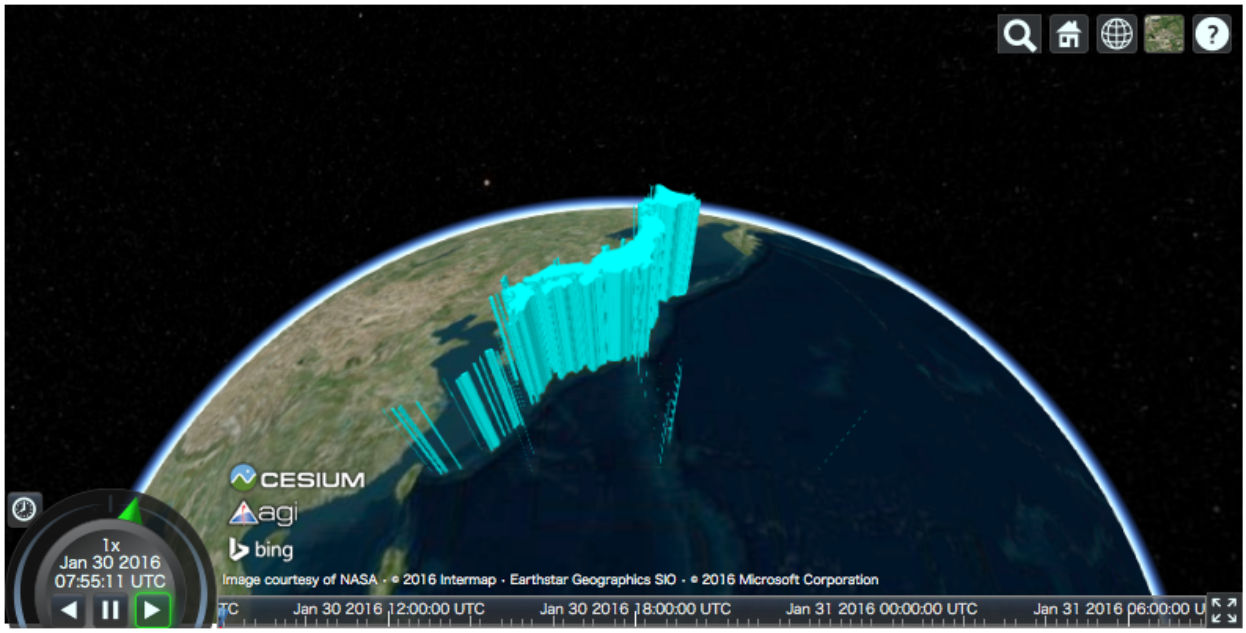
```
>>> res = cesiumpy.io.read_geojson('jpn.geo.json')
>>> type(res)
list
```

You can add the `list` as entities.

```
>>> viewer = cesiumpy.Viewer()
>>> viewer.entities.add(res)
>>> viewer
```

If you want to change some properties, passing keyword arguments via `entities.add` methods is easy. Of cource it is also OK to change properties of each entity one by one.

```
>>> viewer = cesiumpy.Viewer()
>>> viewer.entities.add(res, extrudedHeight=1e6, material='aqua')
>>> viewer
```



4.2.2 Shapefile

This example reads Shapefile of Japanese coastal lines. `cesiumpy.io.read_shape` returns a `list` of `cesiumpy.Polyline`.

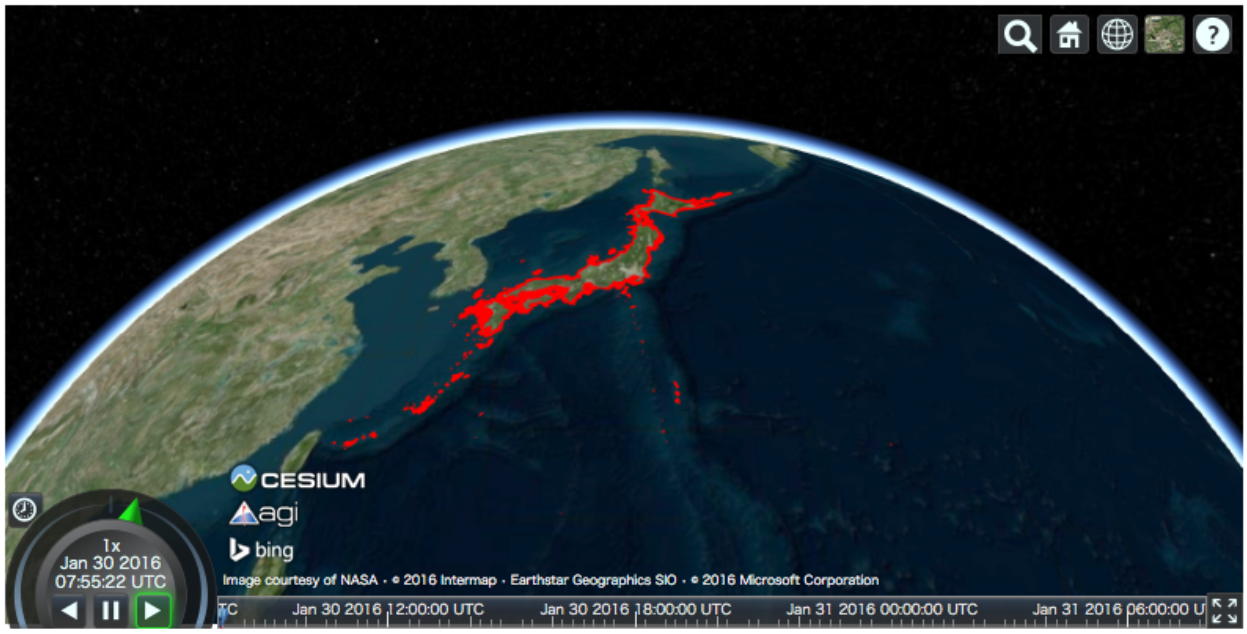
The file is provided by website.

- (Source)

```
>>> res = cesiumpy.io.read_shape('coastl_jpn.shp')
>>> type(res)
list
```

Then, you can add the result to the map.

```
>>> viewer = cesiumpy.Viewer()
>>> viewer.entities.add(res, material='red')
>>> viewer
```

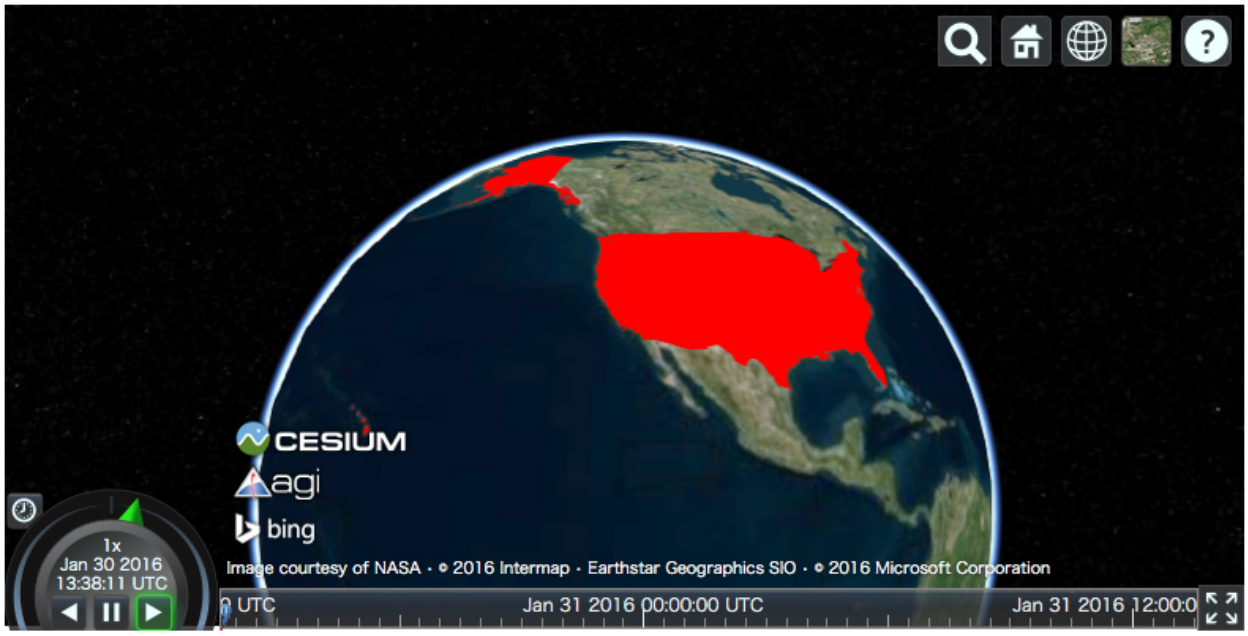


4.3 Bundled Data

`cesiumpy` bundles GeoJSON data provided by mledoze/countries repositry. You can load them via `cesiumpy.countries.get` method passing country code or its name.

Please refer to countries.json file to check available country codes ("cca2" or "cca3") and names ("official name").

```
>>> usa = cesiumpy.countries.get('USA')
>>> viewer = cesiumpy.Viewer()
>>> viewer.entities.add(usa, material='red')
>>> viewer
```



4.4 Read 3D Models

Cesium.js can handle 3D Model on the map. For `Cesium.js` functionality, please refer to 3D Model Tutorial.

`cesiumpy` allows to put 3D Model using `cesiumpy.Model` instance. Following example shows to draw Cesium Man used in the above tutorial.

```
>>> m = cesiumpy.Model(url='data/Cesium_Man.gltf',
...                    modelMatrix=(-130, 40, 0.0), scale=1000000)
>>> m
Model("data/Cesium_Man.gltf")

>>> viewer = cesiumpy.Viewer()
>>> viewer.scene.primitives.add(m)
>>> viewer
```

CHAPTER 5

Examples

This section lists some examples using `cesiumpy` and other packages. You can find `Jupyter Notebook` of these exampless under `GitHub` repository (maps are not rendered on `GitHub`. Download an run them on local).

- https://github.com/sinhrks/cesiumpy/tree/master/examples

5.1 Use with pandas

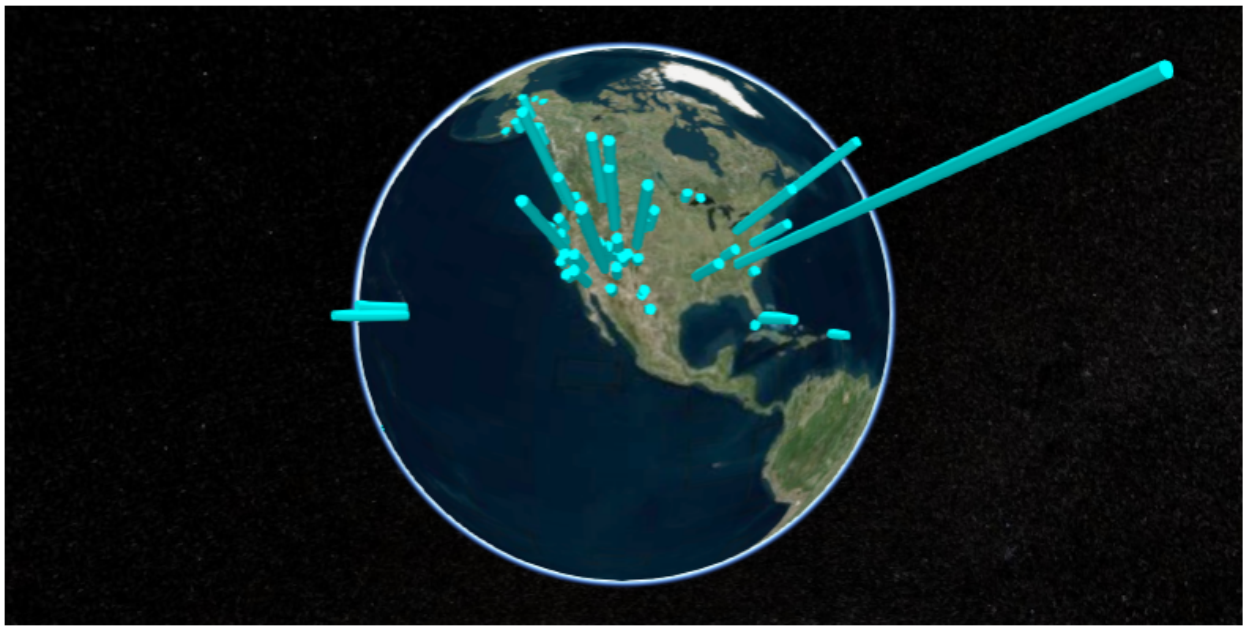
Following example shows retrieving "US National Parks" data from Wikipedia, then plot number of visitors on the map.

First, load data from Wikipedia using `pd.read_html` functionality. The data contains latitude and longitude as text, thus some preprocessing is required.

```
>>> import pandas as pd
>>> url = "https://en.wikipedia.org/wiki/List_of_national_parks_of_the_United_States"
>>> df = pd.read_html(url, header=0)[0]

>>> locations = df['Location'].str.extract(u'(\D+) (\d+°\d+[NS]) (\d+°\d+[WE]).*')
>>> locations.columns = ['State', 'lat', 'lon']
>>> locations['lat'] = locations['lat'].str.replace(u'°', '.')
>>> locations['lon'] = locations['lon'].str.replace(u'°', '.')
>>> locations.loc[locations['lat'].str.endswith('S'), 'lat'] = '-' + locations['lat']
>>> locations.loc[locations['lon'].str.endswith('W'), 'lon'] = '-' + locations['lon']

>>> locations['lat'] = locations['lat'].str.slice_replace(start=-2)
>>> locations['lon'] = locations['lon'].str.slice_replace(start=-2)
>>> locations[['lat', 'lon']] = locations[['lat', 'lon']].astype(float)

>>> locations.head()
            State    lat     lon
0           Maine  44.21  -68.13
1  American Samoa -14.15 -170.41
2            Utah  38.41 -109.34
3    South Dakota  43.45 -102.30
4           Texas  29.15 -103.15

>>> df = pd.concat([df, locations], axis=1)
```

Once prepared the data, iterate over rows and plot its values. The below script adds `cesiumpy.Cylinder` which height is corresponding to the number of visitors.

```
>>> import cesiumpy

>>> options = dict(animation=True, baseLayerPicker=False, fullscreenButton=False,
...                geocoder=False, homeButton=False, infoBox=False, sceneModePicker=True,
...                selectionIndicator=False, navigationHelpButton=False,
...                timeline=False, navigationInstructionsInitiallyVisible=False)

>>> v = cesiumpy.Viewer(**options)

>>> for i, row in df.iterrows():
...     l = row['Recreation Visitors (2014)[5]']
...     cyl = cesiumpy.Cylinder(position=[row['lon'], row['lat'], l / 2.], length=l,
...                             topRadius=10e4, bottomRadius=10e4, material='aqua', alpha=0.5)
...     v.entities.add(cyl)

>>> v
```



If you want bubble chart like output, use `Point` entities.

```
>>> v = cesiumpy.Viewer(**options)
>>> for i, row in df.iterrows():
...     l = row['Recreation Visitors (2014)[5]']
...      p= cesiumpy.Point(position=[row['lon'], row['lat'], 0],
...                        pixelSize=np.sqrt(l / 10000), color='blue')
>>> v.entities.add(p)
>>> v
```

Using `Billboard` with `Pin` points the locations of National Parks.

```
>>> v = cesiumpy.Viewer(**options)
>>> pin = cesiumpy.Pin()
>>> for i, row in df.iterrows():
...     b = cesiumpy.Billboard(position=[row['lon'], row['lat'], 0], image = pin, scale=0.4)
>>> v.entities.add(b)
>>> v
```

5.2 Use with shapely / geopandas

Following example shows how to handle `geojson` files using `shapely`, `geopandas` and `cesiumpy`.

First, read `geojson` file of US, California using `geopandas` function. The content will be `shapely` instance.

```
>>> import geopandas as gpd

>>> df = gpd.read_file('ca.json')
>>> df.head()
  fips                                           geometry      id        name
0   06  POLYGON ((-123.233256 42.006186, -122.378853 4...  USA-CA  California

>>> g = df.loc[0, "geometry"]
>>> type(g)
shapely.geometry.polygon.Polygon
```

We can use this `shapely` instance to specify the shape of `cesiumpy` instances. The below script adds `cesiumpy.Wall` which has the shape of California.

```
>>> import cesiumpy

>>> options = dict(animation=True, baseLayerPicker=False, fullscreenButton=False,
...                geocoder=False, homeButton=False, infoBox=False, sceneModePicker=True,
...                selectionIndicator=False, navigationHelpButton=False,
...                timeline=False, navigationInstructionsInitiallyVisible=False)

>>> v = cesiumpy.Viewer(**options)
>>> v.entities.add(cesiumpy.Wall(positions=g,
...                              maximumHeights=10e5, minimumHeights=0,
...                              material=cesiumpy.color.RED))
>>> v
```

5.3 Use with scipy

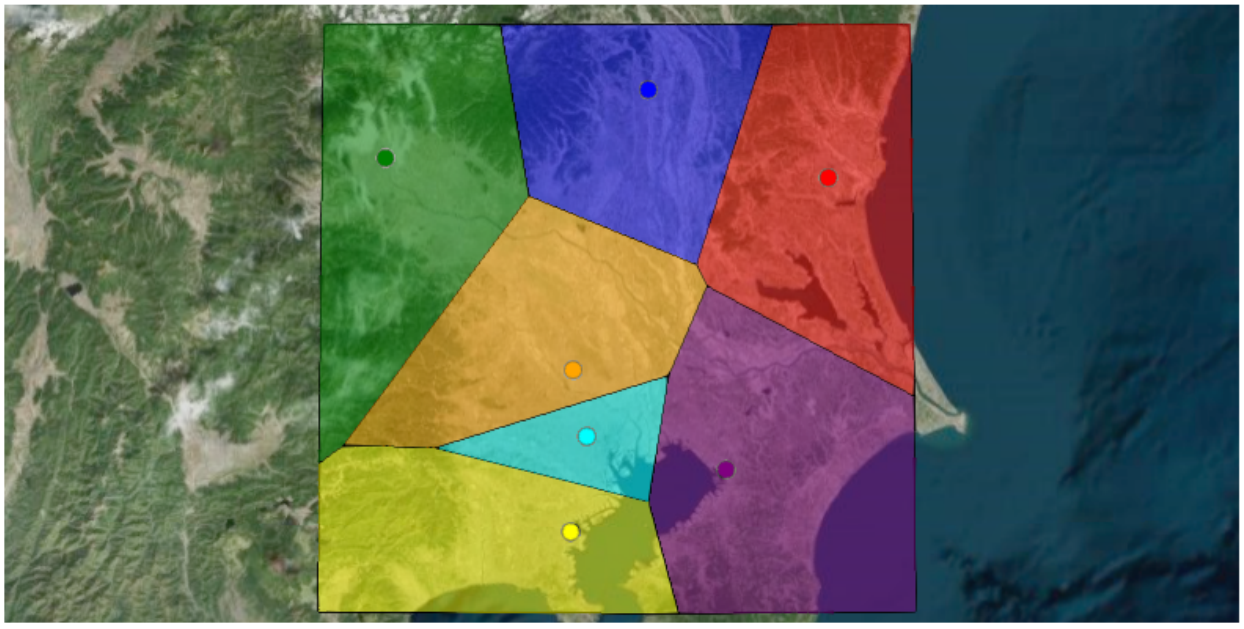
`cesiumpy` has `spatial` submodule which offers functionality like `scipy.spatial`. These function requires `scipy` and `shapely` installed.

Following example shows Voronoi diagram using `cesiumpy`. First, prepare a list contains the geolocations of Japanese Prefectual goverments.

```
>>> import cesiumpy

>>> options = dict(animation=True, baseLayerPicker=False, fullscreenButton=False,
...                geocoder=False, homeButton=False, infoBox=False, sceneModePicker=True,
...                selectionIndicator=False, navigationHelpButton=False,
...                timeline=False, navigationInstructionsInitiallyVisible=False)

>>> points = [[140.446793, 36.341813],
...           [139.883565, 36.565725],
...           [139.060156, 36.391208],
...           [139.648933, 35.857428],
...           [140.123308, 35.605058],
...           [139.691704, 35.689521],
...           [139.642514, 35.447753]]
```

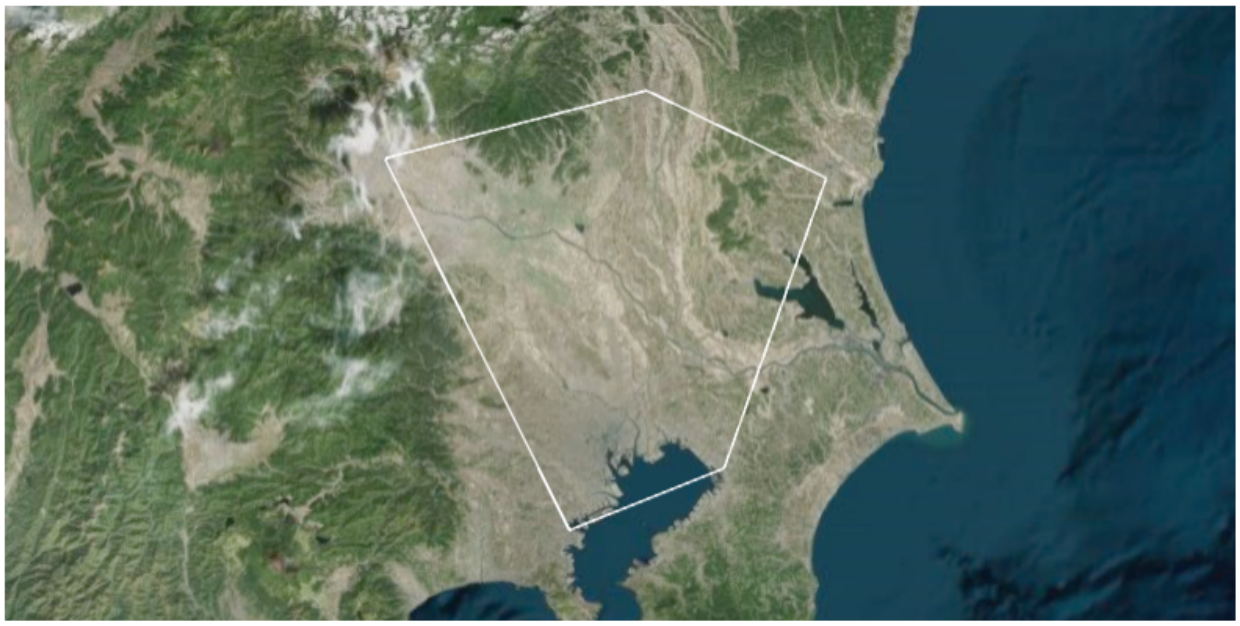
Then, you can create `cesiumpy.spatial.Voronoi` instance passing `points`. Using `get_polygons` method returns the `list` of `cesiumpy.Polygon` instances. Each `Polygon` represents the region corresponding to the point.

```
>>> vor = cesiumpy.spatial.Voronoi(points)
>>> polygons = vor.get_polygons()
>>> polygons[0]
Polygon([140.70970652380953, 35.78698294268851, 140.06610971077615, 36.06956523268194, 140.033676546(

>>> v = cesiumpy.Viewer(**options)
>>> colors = [cesiumpy.color.RED, cesiumpy.color.BLUE, cesiumpy.color.GREEN,
...           cesiumpy.color.ORANGE, cesiumpy.color.PURPLE, cesiumpy.color.AQUA,
...           cesiumpy.color.YELLOW]

>>> for p, pol, c in zip(points, polygons, colors):
...     b = cesiumpy.Point(position=(p[0], p[1], 0), color=c)
...     v.entities.add(b)
...     pol.material = c.set_alpha(0.5)
...     pol.outline = True
...     v.entities.add(pol)
>>> v.camera.flyTo((139.8, 36, 3e5))
>>> v
```



Next example shows to draw convex using `cesiumpy`. You can use `cesiumpy.spatial.ConvexHull` class, then use `get_polyline` method to get the `cesiumpy.Polyline` instances. `Polyline` contains the coordinates of convex.

```
>>> conv = cesiumpy.spatial.ConvexHull(points)
>>> polyline = conv.get_polyline()
>>> polyline
Polyline([139.060156, 36.391208, 139.642514, 35.447753, 140.123308, 35.605058, 140.446793, 36.341813,

>>> v = cesiumpy.Viewer(**options)
>>> v.entities.add(polyline)
>>> v.camera.flyTo((139.8, 36, 3e5))
>>> v
```



API:

CHAPTER 6

cesiumpy package

6.1 Subpackages

6.1.1 cesiumpy.data package

Subpackages

cesiumpy.data.tests package

Submodules

Module contents

Submodules

class cesiumpy.data.country.**CountryLoader**
Bases: object

countries

get(*name*)

Module contents

6.1.2 cesiumpy.entities package

Subpackages

cesiumpy.entities.tests package

Submodules

Module contents

Submodules

class cesiumpy.entities.cartesian.**Cartesian2**(*x*, *y*, *degrees=False*)
Bases: cesiumpy.entities.cartesian._Cartesian

classmethod fromDegrees(*x*, *y*)

classmethod maybe(*x*, *degrees=False*)
Convert list or tuple to Cartesian2

x
A float trait.

y
A float trait.

class cesiumpy.entities.cartesian.**Cartesian3**(*x*, *y*, *z*, *degrees=False*)
Bases: cesiumpy.entities.cartesian._Cartesian

classmethod fromDegrees(*x*, *y*, *z*)

classmethod fromDegreesArray(*x*)

classmethod maybe(*x*, *degrees=False*)
Convert list or tuple to Cartesian3

x
A float trait.

y
A float trait.

z
A float trait.

class cesiumpy.entities.cartesian.**Cartesian3Array**(*x*)
Bases: cesiumpy.entities.cartesian._Cartesian

class cesiumpy.entities.cartesian.**Cartesian4**(*x*, *y*, *z*, *w*, *degrees=False*)
Bases: cesiumpy.entities.cartesian._Cartesian

classmethod fromDegrees(*x*, *y*, *z*, *w*)

classmethod maybe(*x*, *degrees=False*)
Convert list or tuple to Cartesian4

w
A float trait.

x
A float trait.

y
A float trait.

z
A float trait.

class cesiumpy.entities.cartesian.**Rectangle**(*west*, *south*, *east*, *north*, *degrees=False*)
Bases: cesiumpy.entities.cartesian._Cartesian

east
A float trait.

classmethod fromDegrees(*west*, *south*, *east*, *north*)

classmethod maybe(*x*)

north
A float trait.

script

south
A float trait.

west
A float trait.

class cesiumpy.entities.color.**Color**(*red*, *green*, *blue*, *alpha=None*)
Bases: *cesiumpy.entities.material.Material*

alpha
A float trait.

blue
A float trait.

copy()

classmethod fromAlpha(*color*, *alpha*)

classmethod fromBytes(*red=255*, *green=255*, *blue=255*, *alpha=None*)
Creates a new Color specified using red, green, blue, and alpha values that are in the range of 0 to 255, converting them internally to a range of 0.0 to 1.0.

red: int, default 255 The red component.

green: int, default 255 The green component.

blue: int, default 255 The blue component.

alpha: int, default None The alpha component.

classmethod fromCssColorString(*color*)
Creates a Color instance from a CSS color value.

color: str The CSS color value in #rgb, #rrggbb, rgb(), rgba(), hsl(), or hsla() format.

classmethod fromString(*color*)
Creates a Color instance from a CSS color value. Shortcut for Color.fromCssColorString.

color: str The CSS color value in #rgb, #rrggbb, rgb(), rgba(), hsl(), or hsla() format.

green
A float trait.

classmethod maybe(*x*)
Convert str or tuple to ColorConstant

red
A float trait.

script

set_alpha(*alpha*)

withAlpha(*alpha*)

class cesiumpy.entities.color.**ColorConstant**(*name*, *alpha=None*)
Bases: *cesiumpy.entities.color.CssColor*

class cesiumpy.entities.color.**ColorFactory**
Bases: object

Color
return Color class

choice()
Randomly returns a single color.

get_cmap(*name*)

sample(*n*)
Randomly returns list of colors which length is n.

class cesiumpy.entities.color.**ColorMap**(*name*)
Bases: cesiumpy.base._CesiumObject

name
A trait for unicode strings.

class cesiumpy.entities.color.**CssColor**(*name*, *alpha=None*)
Bases: *cesiumpy.entities.color.Color*

alpha
A float trait.

copy()

name
A trait for unicode strings.

script

class cesiumpy.entities.entity.**Billboard**(*position*, *image=None*, *show=None*, *scale=None*, *horizontalOrigin=None*, *verticalOrigin=None*, *eyeOffset=None*, *pixelOffset=None*, *rotation=None*, *alignedAxis=None*, *width=None*, *height=None*, *color=None*, *scaleByDistance=None*, *translucencyByDistance=None*, *pixelOffsetScaleByDistance=None*, *imageSubRegion=None*, *sizeInMeters=None*, *name=None*)
Bases: cesiumpy.entities.entity._CesiumEntity

PointGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

image [str or Pin, default Pin()] A Property specifying the Image, URI, or Canvas to use for the billboard.

show [bool, default True] A boolean Property specifying the visibility of the billboard.

scale [float, default 1.] A numeric Property specifying the scale to apply to the image size.

horizontalOrigin [HorizontalOrigin, default HorizontalOrigin.CENTER] A Property specifying the HorizontalOrigin.

verticalOrigin [VerticalOrigin, default VerticalOrigin.CENTER] A Property specifying the VerticalOrigin.

eyeOffset [Cartesian3, default Cartesian3.ZERO] A Cartesian3 Property specifying the eye offset.

pixelOffset [Cartesian2, default Cartesian2.ZERO] A Cartesian2 Property specifying the pixel offset.

rotation [float, default 0.] A numeric Property specifying the rotation about the alignedAxis.

alignedAxis [Cartesian3, default Cartesian3.ZERO] A Cartesian3 Property specifying the axis of rotation.

width [float] A numeric Property specifying the width of the billboard in pixels, overriding the native size.

height [float] A numeric Property specifying the height of the billboard in pixels, overriding the native size.

color [Color, default Color.WHITE] A Property specifying the tint Color of the image.

scaleByDistance : A NearFarScalar Property used to scale the point based on distance from the camera.

translucencyByDistance : optional A NearFarScalar Property used to set translucency based on distance from the camera.

pixelOffsetScaleByDistance : optional A NearFarScalar Property used to set pixelOffset based on distance from the camera.

imageSubRegion : A Property specifying a BoundingRectangle that defines a sub-region of the image to use for the billboard, rather than the entire image.

sizeInMeters [bool] A boolean Property specifying whether this billboard's size should be measured in meters.

image
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.entities.entity.**Box**(*position*, *dimensions*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

BoxGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

dimensions [Cartesian3] A Cartesian3 Property specifying the length, width, and height of the box.

show [bool, default True] A boolean Property specifying the visibility of the box.

fill [bool, default True] A boolean Property specifying whether the box is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the box.

outline [bool, default False] A boolean Property specifying whether the box is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

dimensions

class cesiumpy.entities.entity.**Corridor**(*positions*, *width*, *cornerType=None*, *height=None*, *extrudedHeight=None*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *granularity=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

CorridorGraphics

positions [Cartesian3] A Property specifying the array of Cartesian3 positions that define the centerline of the corridor.

width [float] A numeric Property specifying the distance between the edges of the corridor.

cornerType [CornerType, default CornerType.ROUNDED] A CornerType Property specifying the style of the corners.

height [float, default 0.] A numeric Property specifying the altitude of the corridor.

extrudedHeight [float, default 0.] A numeric Property specifying the altitude of the corridor extrusion.

show [bool, default True] A boolean Property specifying the visibility of the corridor.

fill [bool, default True] A boolean Property specifying whether the corridor is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the corridor.

outline [bool, default False] A boolean Property specifying whether the corridor is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the distance between each latitude and longitude.

cornerType
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

positions
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.entities.entity.**Cylinder**(*position*, *length*, *topRadius*, *bottomRadius*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *numberOfVerticalLines=None*, *slices=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

CylinderGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

length [float] A numeric Property specifying the length of the cylinder.

topRadius [float] A numeric Property specifying the radius of the top of the cylinder.

bottomRadius [float] A numeric Property specifying the radius of the bottom of the cylinder.

show [bool, default True] A boolean Property specifying the visibility of the cylinder.

fill [bool, default True] A boolean Property specifying whether the cylinder is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the cylinder.

outline [bool, default False] A boolean Property specifying whether the cylinder is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

numberOfVerticalLines [int, default 16] A numeric Property specifying the number of vertical lines to draw along the perimeter for the outline.

slices [int, default 128] The number of edges around perimeter of the cylinder.

bottomRadius
A float trait.

length
A float trait.

slices
A float trait.

topRadius
A float trait.

class cesiumpy.entities.entity.**Ellipse**(*position*, *semiMinorAxis*, *semiMajorAxis*, *height=None*, *extrudedHeight=None*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *numberOfVerticalLines=None*, *rotation=None*, *stRotation=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

EllipseGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

semiMajorAxis [float] The numeric Property specifying the semi-major axis.

semiMinorAxis [float] The numeric Property specifying the semi-minor axis.

height [float, default 0.] A numeric Property specifying the altitude of the ellipse.

extrudedHeight [float, default 0.] A numeric Property specifying the altitude of the ellipse extrusion.

show [bool, default True] A boolean Property specifying the visibility of the ellipse.

fill [bool, default True] A boolean Property specifying whether the ellipse is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the ellipse.

outline [bool, default False] A boolean Property specifying whether the ellipse is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

numberOfVerticalLines [int, default 16] Property specifying the number of vertical lines to draw along the perimeter for the outline.

rotation [float, default 0.] A numeric property specifying the rotation of the ellipse counter-clockwise from north.

stRotation [float, default 0.] A numeric property specifying the rotation of the ellipse texture counter-clockwise from north.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between points on the ellipse.

semiMajorAxis
A float trait.

semiMinorAxis
A float trait.

class cesiumpy.entities.entity.**Ellipsoid**(*position*, *radii*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *subdivisions=None*, *stackPartitions=None*, *slicePartitions=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

EllipsoidGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

radii [Cartesian3] A Cartesian3 Property specifying the radii of the ellipsoid.

show [bool, default True] A boolean Property specifying the visibility of the ellipsoid.

fill [bool, default True] A boolean Property specifying whether the ellipsoid is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the ellipsoid.

outline [bool, default False] A boolean Property specifying whether the ellipsoid is outlined.

outlineColor [CeciumColor, BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

subdivisions [int, default 128] A Property specifying the number of samples per outline ring, determining the granularity of the curvature.

stackPartitions [int, default 64] A Property specifying the number of stacks.

slicePartitions [int, default 64] A Property specifying the number of radial slices.

radii

slicePartitions
A float trait.

stackPartitions
A float trait.

subdivisions
A float trait.

class cesiumpy.entities.entity.**Label**(*position*, *text*, *style=None*, *fillColor=None*, *outlineColor=None*, *outlineWidth=None*, *show=None*, *scale=None*, *horizontalOrigin=None*, *verticalOrigin=None*, *eyeOffset=None*, *pixelOffset=None*, *translucencyByDistance=None*, *pixelOffsetScaleByDistance=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

LabelGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

text [str] A Property specifying the text.

font [str, default '10px sans-serif'] A Property specifying the CSS font.

style [LabeStyle, default LabelStyle.FILL] A Property specifying the LabelStyle.

fillColor [Color, default Color.WHITE] A Property specifying the fill Color.

outlineColor [Color, default, Color.BLACK] A Property specifying the outline Color.

outlineWidth [float, default 1.] A numeric Property specifying the outline width.

show [bool, default True] A boolean Property specifying the visibility of the label.

scale [float, default 1.] A numeric Property specifying the scale to apply to the text.

horizontalOrigin [HorizontalOrigin, default HorizontalOrigin.CENTER] A Property specifying the HorizontalOrigin.

verticalOrigin [VerticalOrigin, default VerticalOrigin.CENTER] A Property specifying the VerticalOrigin.

eyeOffset [Cartesian3, default Cartesian3.ZERO] A Cartesian3 Property specifying the eye offset.

pixelOffset [Cartesian2, default Cartesian2.ZERO] A Cartesian2 Property specifying the pixel offset.

translucencyByDistance : A NearFarScalar Property used to set translucency based on distance from the camera.

pixelOffsetScaleByDistance : A NearFarScalar Property used to set pixelOffset based on distance from the camera.

fillColor

text
A trait for unicode strings.

class cesiumpy.entities.entity.**Point**(*position*, *color=None*, *pixelSize=10*, *outlineColor=None*, *outlineWidth=None*, *show=None*, *scaleByDistance=None*, *translucencyByDistance=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

PointGraphics

position [Cartesian3] A Property specifying the Cartesian3 positions.

color [Color, default WHITE] A Property specifying the Color of the point.

pixelSize [int, default 10] A numeric Property specifying the size in pixels.

outlineColor [Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [int, default 0] A numeric Property specifying the the outline width in pixels.

show [bool, default True] A boolean Property specifying the visibility of the point.

scaleByDistance : A NearFarScalar Property used to scale the point based on distance.

translucencyByDistance : A NearFarScalar Property used to set translucency based on distance from the camera.

pixelSize
A float trait.

class cesiumpy.entities.entity.**Polygon**(*hierarchy*, *height=None*, *extrudedHeight=None*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *stRotation=None*, *granularity=None*, *perPositionHeight=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

PolygonGraphics

hierarchy [Cartesian3] A Property specifying the PolygonHierarchy.

height [float, default 0.] A numeric Property specifying the altitude of the polygon.

extrudedHeight [float, default 0.] A numeric Property specifying the altitude of the polygon extrusion.

show [bool, default True] A boolean Property specifying the visibility of the polygon.

fill [bool, default True] A boolean Property specifying whether the polygon is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the polygon.

outline [bool, default False] A boolean Property specifying whether the polygon is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default cesiumpy.color.BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

stRotation [float, default 0.] A numeric property specifying the rotation of the polygon texture counter-clockwise from north.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between each latitude and longitude point.

perPositionHeight [bool, default False] A boolean specifying whether or not the the height of each position is used.

perPositionHeight
A boolean (True, False) trait.

positions

class cesiumpy.entities.entity.**Polyline**(*positions*, *followSurface=None*, *width=None*, *show=None*, *material=None*, *granularity=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

PolylineGraphics

positions [Cartesian3] A Property specifying the array of Cartesian3 positions that define the line strip.

followSurface [bool, default True] A boolean Property specifying whether the line segments should be great arcs or linearly connected.

width [float, default 1.] A numeric Property specifying the width in pixels.

show [bool, default True] A boolean Property specifying the visibility of the polyline.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to draw the polyline.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between each latitude and longitude if followSurface is true.

followSurface
A boolean (True, False) trait.

positions
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.entities.entity.**PolylineVolume**(*positions*, *shape*, *cornerType=None*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *granularity=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

PolylineVolumeGraphics

positions [Cartesian3] A Property specifying the array of Cartesian3 positions which define the line strip.

shape [Cartesian2] optional A Property specifying the array of Cartesian2 positions which define the shape to be extruded.

cornerType [CornerType, default ROUNDED] A CornerType Property specifying the style of the corners.

show [bool, default True] A boolean Property specifying the visibility of the volume.

fill [bool, default True] A boolean Property specifying whether the volume is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the volume.

outline [bool, default False] A boolean Property specifying whether the volume is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between each latitude and longitude point.

cornerType
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

positions
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

shape
An instance of a Python list.

class cesiumpy.entities.entity.**Rectangle**(*coordinates*, *height=None*, *extrudedHeight=None*, *closeTop=None*, *closeBottom=None*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *stRotation=None*, *granularity=None*, *name=None*)

Bases: cesiumpy.entities.entity._CesiumEntity

RectangleGraphics

coordinates [list of 4 floats, corresponding to west, south, east, north] The Property specifying the Rectangle.

height [float, default 0.] A numeric Property specifying the altitude of the rectangle.

extrudedHeight [float, default 0.] A numeric Property specifying the altitude of the rectangle extrusion.

closeTop [bool, default True] A boolean Property specifying whether the rectangle has a top cover when extruded

closeBottom [bool, default True] A boolean Property specifying whether the rectangle has a bottom cover when extruded.

show [bool, default True] A boolean Property specifying the visibility of the rectangle.

fill [bool, default True] A boolean Property specifying whether the rectangle is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the rectangle.

outline [bool, default False] A boolean Property specifying whether the rectangle is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

rotation [float, default 0.] A numeric property specifying the rotation of the rectangle clockwise from north.

stRotation [float, default 0.] A numeric property specifying the rotation of the rectangle texture counter-clockwise from north.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between points on the rectangle.

closeBottom
A boolean (True, False) trait.

closeTop
A boolean (True, False) trait.

coordinates

class cesiumpy.entities.entity.**Wall**(*positions*, *maximumHeights*, *minimumHeights*, *show=None*, *fill=None*, *material=None*, *outline=None*, *outlineColor=None*, *outlineWidth=None*, *granularity=None*, *name=None*)
Bases: cesiumpy.entities.entity._CesiumEntity

WallGraphics

positions [Cartesian3] A Property specifying the array of Cartesian3 positions which define the top of the wall.

maximumHeights [float or its list] A Property specifying an array of heights to be used for the top of the wall instead of the height of each position.

minimumHeights [float or its list] A Property specifying an array of heights to be used for the bottom of the wall instead of the globe surface.

show [bool, default True] A boolean Property specifying the visibility of the wall.

fill [bool, default True] A boolean Property specifying whether the wall is filled with the provided material.

material [cesiumpy.cesiumpy.color.Color, default WHITE] A Property specifying the material used to fill the wall.

outline [bool, default False] A boolean Property specifying whether the wall is outlined.

outlineColor [cesiumpy.cesiumpy.color.Color, default BLACK] A Property specifying the Color of the outline.

outlineWidth [float, default 1.] A numeric Property specifying the width of the outline.

granularity [float, default cesiumpy.math.RADIANS_PER_DEGREE] A numeric Property specifying the angular distance between each latitude and longitude point.

positions
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.entities.material.**ImageMaterialProperty**(*image*, *repeat=None*)
Bases: *cesiumpy.entities.material.Material*

image [str] A Property specifying the Image, URL, Canvas, or Video.

repeat [Cartesian2, default new Cartesian2(1.0, 1.0)] A Cartesian2 Property specifying the number of times

image

script

class cesiumpy.entities.material.**Material**(**args*, ***kwargs*)
Bases: cesiumpy.base._CesiumObject

classmethod maybe(*x*)

class cesiumpy.entities.material.**TemporaryImage**(*figure*, *trim=True*)
Bases: cesiumpy.base._CesiumObject

Receive an image and output a temp file

figure [matplotib Figure or Axes] Instance to be drawn as an image. When trim is True, figure should only contain a single Axes.

trim [bool, default True] Whether to trim margins of

path
A trait for unicode strings.

script

trim
A boolean (True, False) trait.

class cesiumpy.entities.model.**Model**(*url*, *modelMatrix*, *basePath=None*, *show=None*, *scale=None*, *minimumPixelSize=None*, *maximumScale=None*, *id=None*, *allowPicking=None*, *incrementallyLoadTextures=None*, *asynchronous=None*, *debugShowBoundingVolume=None*, *debugWireframe=None*)
Bases: cesiumpy.base._CesiumObject

3D Model

url [str] The object for the glTF JSON or an arraybuffer of Binary glTF defined by the KHR_binary_glTF extension.

modelMatrix [Matrix4, default Matrix4.IDENTITY] The 4x4 transformation matrix that transforms the model from model to world coordinates.

basePath [str, default ‘’] The base path that paths in the glTF JSON are relative to.

show [bool, default True] Determines if the model primitive will be shown.

scale [float, default 1.0] A uniform scale applied to this model.

minimumPixelSize [float, default 0.0] The approximate minimum pixel size of the model regardless of zoom.

maximumScale [float] The maximum scale size of a model. An upper limit for minimumPixelSize.

id : A user-defined object to return when the model is picked with Scene#pick.

allowPicking [bool, default True] When true, each glTF mesh and primitive is pickable with Scene#pick.

incrementallyLoadTextures [bool, default True] Determine if textures may continue to stream in after the model is loaded.

asynchronous [bool, default True] Determines if model WebGL resource creation will be spread out over several frames or block until completion once all glTF files are loaded.

debugShowBoundingVolume [bool, default False] For debugging only. Draws the bounding sphere for each draw command in the model.

debugWireframe [bool, default False] For debugging only. Draws the model in wireframe.

allowPicking
A boolean (True, False) trait.

asynchronous
A boolean (True, False) trait.

basePath
A trait for unicode strings.

debugShowBoundingVolume
A boolean (True, False) trait.

debugWireframe
A boolean (True, False) trait.

incrementallyLoadTextures
A boolean (True, False) trait.

maximumScale
A float trait.

minimumPixelSize
A float trait.

modelMatrix
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

scale
A float trait.

script

show
A boolean (True, False) trait.

url

class cesiumpy.entities.pinbuilder.**Icon**(*image*)
Bases: cesiumpy.entities.pinbuilder._BillboardContents

image

script

class cesiumpy.entities.pinbuilder.**Pin**(*color=None*, *size=48*, *text=None*)
Bases: cesiumpy.entities.pinbuilder._BillboardContents

color

classmethod fromColor(*color*, *size=48*)
Create pin specifying color and size

color [Color] A Property specifying the Color of the pin.

size [int, default 48] A Property specifying the size of the pin.

classmethod fromText(*text*, *color=None*, *size=48*)
Create pin specifying text, color and size

text [str] A Property specifying the text of the pin.

color [Color] A Property specifying the Color of the pin.

size [int, default 48] A Property specifying the size of the pin.

script

size
A float trait.

text
A trait for unicode strings.

class cesiumpy.entities.transform.**Transforms**(*origin*, *transform*)
Bases: cesiumpy.base._CesiumObject

classmethod eastNorthUpToFixedFrame(*origin*)
Computes a 4x4 transformation matrix from a reference frame with an east-north-up axes centered at the provided origin to the provided ellipsoid's fixed reference frame. The local axes are defined as:

•The x axis points in the local east direction.

•The y axis points in the local north direction.

•The z axis points in the direction of the ellipsoid surface normal

which passes through the position.

origin [Cartesian3] The center point of the local reference frame.

classmethod northEastDownToFixedFrame(*origin*)
Computes a 4x4 transformation matrix from a reference frame with an north-east-down axes centered at the provided origin to the provided ellipsoid's fixed reference frame. The local axes are defined as:

•The x axis points in the local north direction.

•The y axis points in the local east direction.

•The z axis points in the opposite direction of the ellipsoid surface

normal which passes through the position.

origin [Cartesian3] The center point of the local reference frame.

classmethod northUpEastToFixedFrame(*origin*)
Computes a 4x4 transformation matrix from a reference frame with an north-up-east axes centered at the provided origin to the provided ellipsoid's fixed reference frame. The local axes are defined as:

•The x axis points in the local north direction.

•The y axis points in the direction of the ellipsoid surface normal which passes through the position.

•The z axis points in the local east direction.

origin [Cartesian3] The center point of the local reference frame.

origin

script

transform
A trait for unicode strings.

Module contents

6.1.3 cesiumpy.extension package

Subpackages

cesiumpy.extension.tests package

Submodules

Module contents

Submodules

cesiumpy.extension.io.**read_geojson**(*path*)

cesiumpy.extension.io.**read_shape**(*path*)

class cesiumpy.extension.shapefile.**DummyClass**
Bases: object

cesiumpy.extension.shapefile.**ShapelyLineString**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyLinearRing**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyMultiLineString**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyMultiPoint**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyMultiPolygon**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyPoint**
alias of *DummyClass*

cesiumpy.extension.shapefile.**ShapelyPolygon**
alias of *DummyClass*

cesiumpy.extension.shapefile.**to_entity**(*shape*)
Convert shapely.geometry to corresponding entities. Result may be a list if geometry is consists from multiple instances.

class cesiumpy.extension.spatial.**ConvexHull**(*hull*)
Bases: cesiumpy.extension.spatial._Spatial

Wrapper for scipy.spatial.ConvexHull

hull : ConvexHull of list of points

get_polyline()

class cesiumpy.extension.spatial.**Voronoi**(*vor*)
Bases: cesiumpy.extension.spatial._Spatial

Wrapper for scipy.spatial.Voronoi

vor : Voronoi of list of points

get_polygons()

Module contents

6.1.4 cesiumpy.plotting package

Subpackages

cesiumpy.plotting.tests package

Submodules

Module contents

Submodules

class cesiumpy.plotting.plot.**PlottingAccessor**(*widget*)

Bases: object

bar(*x*, *y*, *z*, *size=10000.0*, *color=None*, *bottom=0.0*)

Plot cesiumpy.Cylinder like bar plot

x [list] List of longitudes

y [list] List of latitudes

z [list] List of bar heights

size [list or float, default 10e3] Radius of cylinder

color [list or Color] Cylinder color

bottom [list or float, default 0] Bottom heights

contour(*x*, *y*, *z*)

Plot contours using cesiumpy.Polyline

X : np.ndarray Y : np.ndarray Y : np.ndarray

X and *Y* must both be 2-D with the same shape as *Z*, or they must both be 1-D such that `len(X)` is the number of columns in *Z* and `len(Y)` is the number of rows in *Z*.

label(*text*, *x*, *y*, *z=None*, *size=None*, *color=None*)

Plot cesiumpy.Label

text [list] List of labels

x [list] List of longitudes

y [list] List of latitudes

z [list or float] Heights

size [list or float] Text size

color [list or Color] Text color

pin(*x*, *y*, *z=None*, *text=None*, *size=None*, *color=None*)

Plot cesiumpy.Pin

x [list] List of longitudes

y [list] List of latitudes

z [list or float] Heights

text [list] List of labels

size [list or float] Text size

color [list or Color] Text color

scatter(*x*, *y*, *z=None*, *size=None*, *color=None*)
Plot cesiumpy.Point like scatter plot

x [list] List of longitudes

y [list] List of latitudes

z [list or float] Height

size [list or float] Pixel size

color [list or Color] Point color

Module contents

6.1.5 cesiumpy.tests package

Submodules

Module contents

6.1.6 cesiumpy.util package

Subpackages

cesiumpy.util.tests package

Submodules

class cesiumpy.util.tests.test_html.**TestHTML**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_add_indent()

test_wrap_script()

test_wrap_uri()

class cesiumpy.util.tests.test_trait.**TestTrait**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_div()

Module contents

Submodules

cesiumpy.util.common.**is_latitude**(*x*)

cesiumpy.util.common.**is_listlike**(*x*)
whether the input can be regarded as list

cesiumpy.util.common.**is_listlike_2elem**(*x*)

cesiumpy.util.common.**is_listlike_3elem**(*x*)

cesiumpy.util.common.**is_longitude**(*x*)

cesiumpy.util.common.**is_numeric**(*x*)

cesiumpy.util.common.**notimplemented**(*x*)

cesiumpy.util.common.**to_jsobject**(*x*)
convert x to JavaScript Object

cesiumpy.util.common.**to_jsscalar**(*x*)
convert x to JavaScript representation

cesiumpy.util.common.**validate_latitude**(*x*, *key*)
validate whether x is numeric, and between -90 and 90

cesiumpy.util.common.**validate_listlike**(*x*, *key*)
validate whether x is list-likes

cesiumpy.util.common.**validate_listlike_even**(*x*, *key*)
validate whether x is list-likes which length is even-number

cesiumpy.util.common.**validate_listlike_lonlat**(*x*, *key*)
validate whether x is list-likes consists from lon, lat pairs

cesiumpy.util.common.**validate_longitude**(*x*, *key*)
validate whether x is numeric, and between -180 and 180

cesiumpy.util.common.**validate_numeric**(*x*, *key*)
validate whether x is int, long or float

cesiumpy.util.common.**validate_numeric_or_none**(*x*, *key*)
validate whether x is int, long, float or None

class cesiumpy.util.trait.**MaybeTrait**(*klass=None*, *args=None*, *kw=None*, ***kwargs*)
Bases: traitlets.traitlets.Instance

validate(*obj*, *value*)

class cesiumpy.util.trait.**URITrait**(*default_value=traitlets.Undefined*, *allow_none=False*, *read_only=None*, *help=None*, ***kwargs*)
Bases: traitlets.traitlets.Unicode

validate(*obj*, *value*)

Module contents

6.2 Submodules

class cesiumpy.base.**RistrictedList**(*widget*, *allowed*, *propertyname*)
Bases: cesiumpy.base._CesiumObject

add(*item*, ***kwargs*)

clear()

script
return list of scripts built from entities each script may be a list of comamnds also

widget
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.camera.**Camera**(*widget*)
Bases: cesiumpy.base._CesiumObject

destination
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

flyTo(*destination*, *orientation=None*)

class cesiumpy.constants.**CornerType**
Bases: cesiumpy.base._CesiumEnum

BEVELED = <CornerType.BEVELED: u'Cesium.CornerType.BEVELED'>

MITERED = <CornerType.MITERED: u'Cesium.CornerType.MITERED'>

ROUNDED = <CornerType.ROUNDED: u'Cesium.CornerType.ROUNDED'>

class cesiumpy.constants.**HorizontalOrigin**
Bases: cesiumpy.base._CesiumEnum

CENTER = <HorizontalOrigin.CENTER: u'Cesium.HorizontalOrigin.CENTER'>

LEFT = <HorizontalOrigin.LEFT: u'Cesium.HorizontalOrigin.LEFT'>

RIGHT = <HorizontalOrigin.RIGHT: u'Cesium.HorizontalOrigin.RIGHT'>

class cesiumpy.constants.**Math**
Bases: cesiumpy.base._CesiumEnum

DEGREES_PER_RADIAN = <Math.RADIANS_PER_DEGREE: u'Cesium.Math.RADIANS_PER_DEGREE'>

EPSILON1 = <Math.EPSILON1: u'Cesium.Math.EPSILON1'>

EPSILON10 = <Math.EPSILON10: u'Cesium.Math.EPSILON10'>

EPSILON11 = <Math.EPSILON11: u'Cesium.Math.EPSILON11'>

EPSILON12 = <Math.EPSILON12: u'Cesium.Math.EPSILON12'>

EPSILON13 = <Math.EPSILON13: u'Cesium.Math.EPSILON13'>

EPSILON14 = <Math.EPSILON14: u'Cesium.Math.EPSILON14'>

EPSILON15 = <Math.EPSILON15: u'Cesium.Math.EPSILON15'>

EPSILON16 = <Math.EPSILON16: u'Cesium.Math.EPSILON16'>

EPSILON17 = <Math.EPSILON17: u'Cesium.Math.EPSILON17'>

EPSILON18 = <Math.EPSILON18: u'Cesium.Math.EPSILON18'>

EPSILON19 = <Math.EPSILON19: u'Cesium.Math.EPSILON19'>

EPSILON2 = <Math.EPSILON2: u'Cesium.Math.EPSILON2'>

EPSILON20 = <Math.EPSILON20: u'Cesium.Math.EPSILON20'>

EPSILON3 = <Math.EPSILON3: u'Cesium.Math.EPSILON3'>

EPSILON4 = <Math.EPSILON4: u'Cesium.Math.EPSILON4'>

EPSILON5 = <Math.EPSILON5: u'Cesium.Math.EPSILON5'>

EPSILON6 = <Math.EPSILON6: u'Cesium.Math.EPSILON6'>

EPSILON7 = <Math.EPSILON7: u'Cesium.Math.EPSILON7'>

EPSILON8 = <Math.EPSILON8: u'Cesium.Math.EPSILON8'>

EPSILON9 = <Math.EPSILON9: u'Cesium.Math.EPSILON9'>

GRAVITATIONALPARAMETER = <Math.GRAVITATIONALPARAMETER: u'Cesium.Math.GRAVITATIONALPARAM

LUNAR_RADIUS = <Math.LUNAR_RADIUS: u'Cesium.Math.LUNAR_RADIUS'>

ONE_OVER_PI = <Math.ONE_OVER_PI: u'Cesium.Math.ONE_OVER_PI'>

ONE_OVER_TWO_PI = <Math.ONE_OVER_TWO_PI: u'Cesium.Math.ONE_OVER_TWO_PI'>

PI = <Math.PI: u'Cesium.Math.PI'>

PI_OVER_FOUR = <Math.PI_OVER_FOUR: u'Cesium.Math.PI_OVER_FOUR'>

PI_OVER_SIX = <Math.PI_OVER_SIX: u'Cesium.Math.PI_OVER_SIX'>

PI_OVER_THREE = <Math.PI_OVER_THREE: u'Cesium.Math.PI_OVER_THREE'>

PI_OVER_TWO = <Math.PI_OVER_TWO: u'Cesium.Math.PI_OVER_TWO'>

RADIANS_PER_ARCSECOND = <Math.RADIANS_PER_ARCSECOND: u'RADIANS_PER_ARCSECOND'>

RADIANS_PER_DEGREE = <Math.RADIANS_PER_DEGREE: u'Cesium.Math.RADIANS_PER_DEGREE'>

SIXTY_FOUR_KILOBYTES = <Math.SIXTY_FOUR_KILOBYTES: u'Cesium.Math.SIXTY_FOUR_KILOBYTES'>

SOLAR_RADIUS = <Math.SOLAR_RADIUS: u'Cesium.Math.SOLAR_RADIUS'>

THREE_PI_OVER_TWO = <Math.THREE_PI_OVER_TWO: u'Cesium.Math.THREE_PI_OVER_TWO'>

TWO_PI = <Math.TWO_PI: u'Cesium.Math.TWO_PI'>

class cesiumpy.constants.**VerticalOrigin**
Bases: cesiumpy.base._CesiumEnum

BOTTOM = <VerticalOrigin.BOTTOM: u'Cesium.VerticalOrigin.BOTTOM'>

CENTER = <VerticalOrigin.CENTER: u'Cesium.VerticalOrigin.CENTER'>

TOP = <VerticalOrigin.TOP: u'Cesium.VerticalOrigin.TOP'>

class cesiumpy.datasource.**CustomDataSource**(*sourceUri*)
Bases: *cesiumpy.datasource.DataSource*

class cesiumpy.datasource.**CzmlDataSource**(*sourceUri*)
Bases: *cesiumpy.datasource.DataSource*

class cesiumpy.datasource.**DataSource**(*sourceUri*)
Bases: cesiumpy.base._CesiumObject

classmethod load(*sourceUri*, **args*, ***kwargs*)

script

sourceUri

class cesiumpy.datasource.**GeoJsonDataSource**(*sourceUri*, *describe=None*, *markerSize=None*, *markerSymbol=None*, *markerColor=None*, *stroke=None*, *strokeWidth=None*, *fill=None*)

Bases: *cesiumpy.datasource.DataSource*

sourceUri [str] Overrides the url to use for resolving relative links.

describe [GeoJsonDataSource~describe, default GeoJsonDataSource.defaultDescribeProperty] A function which returns a Property object (or just a string), which converts the properties into an html description.

markerSize [int, default GeoJsonDataSource.markerSize] The default size of the map pin created for each point, in pixels.

markerSymbol [str, default GeoJsonDataSource.markerSymbol] The default symbol of the map pin created for each point.

markerColor [Color, default GeoJsonDataSource.markerColor] The default color of the map pin created for each point.

stroke [Color, default GeoJsonDataSource.stroke] The default color of polylines and polygon outlines.

strokeWidth [int, GeoJsonDataSource.strokeWidth] The default width of polylines and polygon outlines.

fill [Color, default GeoJsonDataSource.fill] The default color for polygon interiors.

fill

markerColor

markerSize
A float trait.

markerSymbol
A trait for unicode strings.

stroke

strokeWidth
A float trait.

class cesiumpy.datasource.**KmlDataSource**(*sourceUri*)

Bases: *cesiumpy.datasource.DataSource*

sourceUri [str] Overrides the url to use for resolving relative links and other KML network features.

class cesiumpy.provider.**ArcGisImageServerTerrainProvider**(*url*, *token*, *proxy=None*, *tilingScheme=None*, *ellipsoid=None*, *credit=None*)

Bases: *cesiumpy.provider.TerrainProvider*

url [str] The URL of the ArcGIS ImageServer service.

token [str] The authorization token to use to connect to the service.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme specifying how the terrain is broken into tiles. If this parameter is not provided, a GeographicTilingScheme is used.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

credit [Credit or str] The credit, which will is displayed on the canvas.

token
A trait for unicode strings.

class cesiumpy.provider.**ArcGisMapServerImageryProvider**(*url*, *token=None*, *usePreCachedTilesIfAvailable=None*, *layers=None*, *enablePickFeatures=None*, *rectangle=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*, *tileDiscardPolicy=None*, *minimumLevel=None*, *proxy=None*)

Bases: *cesiumpy.provider.ImageryProvider*

ArcGisImageServerTerrainProvider

url [str] The URL of the ArcGIS MapServer service.

token [str] The ArcGIS token used to authenticate with the ArcGIS MapServer service.

usePreCachedTilesIfAvailable [bool, default True] If true, the server's pre-cached tiles are used if they are available. If false, any pre-cached tiles are ignored and the 'export' service is used.

layers [str] A comma-separated list of the layers to show, or undefined if all layers should be shown.

enablePickFeatures [bool, default True] If true, ArcGisMapServerImageryProvider#pickFeatures will invoke the Identify service on the MapServer and return the features included in the response. If false, ArcGisMapServerImageryProvider#pickFeatures will immediately return undefined (indicating no pickable features) without communicating with the server. Set this property to false if you don't want this provider's features to be pickable.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle of the layer. This parameter is ignored when accessing a tiled layer.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme to use to divide the world into tiles. This parameter is ignored when accessing a tiled server.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified and used, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

tileWidth [int, default 256] The width of each tile in pixels. This parameter is ignored when accessing a tiled server.

tileHeight [int, default 256] The height of each tile in pixels. This parameter is ignored when accessing a tiled server.

tileDiscardPolicy [TileDiscardPolicy] The policy that determines if a tile is invalid and should be discarded. If this value is not specified, a default DiscardMissingTileImagePolicy is used for tiled map servers, and a NeverTileDiscardPolicy is used for non-tiled map servers. In the former case, we request tile 0,0 at the maximum tile level and check pixels (0,0), (200,20), (20,200), (80,110), and (160, 130). If all of these pixels are transparent, the discard check is disabled and no tiles are discarded. If any of them have a non-transparent color, any tile that has the same values in these pixel locations is discarded. The end result of these defaults should be correct tile discarding for a standard ArcGIS Server. To ensure that no tiles are discarded, construct and pass a NeverTileDiscardPolicy for this parameter.

maximumLevel [int] The maximum tile level to request, or undefined if there is no maximum. This parameter is ignored when accessing a tiled server.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

enablePickFeatures
A boolean (True, False) trait.

layers
A trait for unicode strings.

token
A trait for unicode strings.

usePreCachedTilesIfAvailable
A boolean (True, False) trait.

class cesiumpy.provider.**BingMapsImageryProvider**(*url*, *key*, *tileProtocol*, *mapStyle=None*, *culture=None*, *ellipsoid=None*, *tileDiscardPolicy=None*, *proxy=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str] The url of the Bing Maps server hosting the imagery.

key [str] The Bing Maps key for your application, which can be created at https://www.bingmapsportal.com/. If this parameter is not provided, BingMapsApi.defaultKey is used. If BingMapsApi.defaultKey is undefined as well, a message is written to the console reminding you that you must create and supply a Bing Maps key as soon as possible. Please do not deploy an application that uses Bing Maps imagery without creating a separate key for your application.

tileProtocol [str] The protocol to use when loading tiles, e.g. 'http:' or 'https:'. By default, tiles are loaded using the same protocol as the page.

mapStyle [str, default BingMapsStyle.AERIAL] The type of Bing Maps imagery to load.

culture [str, default ''] The culture to use when requesting Bing Maps imagery. Not all cultures are supported. See http://msdn.microsoft.com/en-us/library/hh441729.aspx for information on the supported cultures.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

tileDiscardPolicy [TileDiscardPolicy] The policy that determines if a tile is invalid and should be discarded. If this value is not specified, a default DiscardMissingTileImagePolicy is used which requests tile 0,0 at the maximum tile level and checks pixels (0,0), (120,140), (130,160), (200,50), and (200,200). If all of these pixels are transparent, the discard check is disabled and no tiles are discarded. If any of them have a non-transparent color, any tile that has the same values in these pixel locations is discarded. The end result of these defaults should be correct tile discarding for a standard Bing Maps server. To ensure that no tiles are discarded, construct and pass a NeverTileDiscardPolicy for this parameter.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

culture
A trait for unicode strings.

key
A trait for unicode strings.

mapStyle
A trait for unicode strings.

tileProtocol
A trait for unicode strings.

class cesiumpy.provider.**CesiumTerrainProvider**(*url*, *proxy=None*, *requestVertexNormals=None*, *requestWaterMask=None*, *ellipsoid=None*, *credit=None*)

Bases: *cesiumpy.provider.TerrainProvider*

url [str] The URL of the Cesium terrain server.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

requestVertexNormals [bool, default False] Flag that indicates if the client should request additional lighting information from the server, in the form of per vertex normals if available.

requestWaterMask [bool, default False] Flag that indicates if the client should request per tile water masks from the server, if available.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

requestVertexNormals
A boolean (True, False) trait.

requestWaterMask
A boolean (True, False) trait.

class cesiumpy.provider.**EllipsoidTerrainProvider**(*tilingScheme=None*, *ellipsoid=None*)
Bases: *cesiumpy.provider.TerrainProvider*

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme specifying how the ellipsoidal surface is broken into tiles. If this parameter is not provided, a GeographicTilingScheme is used.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

url
A trait for unicode strings.

class cesiumpy.provider.**GoogleEarthImageryProvider**(*url*, *channel*, *path=None*, *ellipsoid=None*, *tileDiscardPolicy=None*, *maximumLevel=None*, *proxy=None*)
Bases: *cesiumpy.provider.ImageryProvider*

url [str] The url of the Google Earth server hosting the imagery.

channel [int] The channel (id) to be used when requesting data from the server. The channel number can be found by looking at the json file located at: earth.localdomain/default_map/query?request=Json&vars=geeServerDefs The /default_map path may differ depending on your Google Earth Enterprise server configuration. Look for the "id" that is associated with a "ImageryMaps" requestType. There may be more than one id available. Example: { layers: [{ id: 1002, requestType: "ImageryMaps" }, { id: 1007, requestType: "VectorMapsRaster" }] }

path [str, default "/default_map"] The path of the Google Earth server hosting the imagery.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

tileDiscardPolicy [TileDiscardPolicy] The policy that determines if a tile is invalid and should be discarded. To ensure that no tiles are discarded, construct and pass a NeverTileDiscardPolicy for this parameter.

maximumLevel [int] The maximum level-of-detail supported by the Google Earth Enterprise server, or undefined if there is no limit.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

channel
A float trait.

path
A trait for unicode strings.

class cesiumpy.provider.**GridImageryProvider**
Bases: *cesiumpy.provider.ImageryProvider*

class cesiumpy.provider.**ImageryProvider**(*url=None*, *fileExtension=None*, *rectangle=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*, *tileDiscardPolicy=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*, *subdomains=None*)
Bases: cesiumpy.provider._CesiumProvider

credit
A trait for unicode strings.

fileExtension
A trait for unicode strings.

maximumLevel
A float trait.

minimumLevel
A float trait.

rectangle

tileHeight
A float trait.

tileWidth
A float trait.

url
A trait for unicode strings.

class cesiumpy.provider.**MapboxImageryProvider**(*url*, *mapId*, *accessToken*, *format=None*, *rectangle=None*, *ellipsoid=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*)
Bases: *cesiumpy.provider.ImageryProvider*

url [str, default '//api.mapbox.com/v4/'] The Mapbox server url.

mapId [str] The Mapbox Map ID.

accessToken [str] The public access token for the imagery.

format [str, default 'png'] The format of the image request.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle, in radians, covered by the image.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

minimumLevel [int, default 0] The minimum level-of-detail supported by the imagery provider. Take care when specifying this that the number of tiles at the minimum level is small, such as four or less. A larger number is likely to result in rendering problems.

maximumLevel [int, default 0] The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL.

accessToken
A trait for unicode strings.

format
A trait for unicode strings.

mapId
A trait for unicode strings.

url
A trait for unicode strings.

class cesiumpy.provider.**OpenStreetMapImageryProvider**(*url=None*, *fileExtension=None*, *rectangle=None*, *ellipsoid=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str, default '//a.tile.openstreetmap.org'] The OpenStreetMap server url.

fileExtension [str, default 'png'] The file extension for images on the server.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle of the layer.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

minimumLevel [int, default 0] The minimum level-of-detail supported by the imagery provider.

maximumLevel [int] The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit.

credit [Credit or str, default 'MapQuest, Open Street Map and contributors, CC-BY-SA'] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL.

class cesiumpy.provider.**SingleTileImageryProvider**(*url*, *rectangle=None*, *ellipsoid=None*, *credit=None*, *proxy=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str] The url for the tile.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle, in radians, covered by the image.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

class cesiumpy.provider.**TerrainProvider**(*url=None*, *proxy=None*, *tilingScheme=None*, *ellipsoid=None*, *credit=None*)

Bases: cesiumpy.provider._CesiumProvider

credit
A trait for unicode strings.

url
A trait for unicode strings.

class cesiumpy.provider.**TileCoordinatesImageryProvider**(*color=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*)

Bases: *cesiumpy.provider.ImageryProvider*

color [cesiumpy.color.Color, default YELLOW] The color to draw the tile box and label.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme for which to draw tiles.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

tileWidth [int, default 256] The width of the tile for level-of-detail selection purposes.

tileHeight [int, default 256] The height of the tile for level-of-detail selection purposes.

class cesiumpy.provider.**TileMapServiceImageryProvider**(*url=None*, *fileExtension=None*, *rectangle=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str, default '.'] Path to image tiles on server.

fileExtension [default 'png'] The file extension for images on the server.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle, in radians, covered by the image.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme specifying how the ellipsoidal surface is broken into tiles. If this parameter is not provided, a WebMercatorTilingScheme is used.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

tileWidth [int, default 256] Pixel width of image tiles.

tileHeight [int, default 256] Pixel height of image tiles.

minimumLevel [int, default 0] The minimum level-of-detail supported by the imagery provider. Take care when specifying this that the number of tiles at the minimum level is small, such as four or less. A larger number is likely to result in rendering problems.

maximumLevel [int] The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit.

credit [Credit or str, default ''] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL.

class cesiumpy.provider.**UrlTemplateImageryProvider**

Bases: *cesiumpy.provider.ImageryProvider*

class cesiumpy.provider.**VRTheWorldTerrainProvider**(*url*, *proxy=None*, *ellipsoid=None*, *credit=None*)

Bases: *cesiumpy.provider.TerrainProvider*

url [str] The URL of the VR-TheWorld TileMap.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

ellipsoid [Ellipsoid, default Ellipsoid.WGS84] The ellipsoid. If this parameter is not specified, the WGS84 ellipsoid is used.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

class cesiumpy.provider.**WebMapServiceImageryProvider**(*url*, *layers*, *parameters=None*, *getFeatureInfoParameters=None*, *enablePickFeatures=None*, *getFeatureInfoFormats=None*, *rectangle=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*, *tileDiscardPolicy=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*, *subdomains=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str] The URL of the WMS service. The URL supports the same keywords as the UrlTemplateImageryProvider.

layers [str] The layers to include, separated by commas.

parameters [Object, default WebMapServiceImageryProvider.DefaultParameters] Additional parameters to pass to the WMS server in the GetMap URL.

getFeatureInfoParameters [Object, default WebMapServiceImageryProvider.GetFeatureInfoDefaultParameters] Additional parameters to pass to the WMS server in the GetFeatureInfo URL.

enablePickFeatures [bool, default True] If true, WebMapServiceImageryProvider#pickFeatures will invoke the GetFeatureInfo operation on the WMS server and return the features included in the response. If false, WebMapServiceImageryProvider#pickFeatures will immediately return undefined (indicating no pickable features) without communicating with the server. Set this property to false if you know your WMS server does not support GetFeatureInfo or if you don't want this provider's features to be pickable.

getFeatureInfoFormats [list of GetFeatureInfoFormat, default WebMapServiceImageryProvider.DefaultGetFeatureInfoFormats] The formats in which to try WMS GetFeatureInfo requests.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle of the layer.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme to use to divide the world into tiles.

ellipsoid [Ellipsoid] The ellipsoid. If the tilingScheme is specified, this parameter is ignored and the tiling scheme's ellipsoid is used instead. If neither parameter is specified, the WGS84 ellipsoid is used.

tileWidth [int, default 256] The width of each tile in pixels.

tileHeight [int, default 256] The height of each tile in pixels.

minimumLevel [int, default 0] The minimum level-of-detail supported by the imagery provider. Take care when specifying this that the number of tiles at the minimum level is small, such as four or less. A larger number is likely to result in rendering problems.

maximumLevel [int] The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit. If not specified, there is no limit.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL, if needed.

subdomains : str or list of str, default 'abc'

enablePickFeatures
A boolean (True, False) trait.

layers
A trait for unicode strings.

class cesiumpy.provider.**WebMapTileServiceImageryProvider**(*url*, *layer*, *style*, *format=None*, *tileMatrixSetID=None*, *tileMatrixLabels=None*, *rectangle=None*, *tillingScheme=None*, *ellipsoid=None*, *tileWidth=None*, *tileHeight=None*, *tileDiscardPolicy=None*, *minimumLevel=None*, *maximumLevel=None*, *credit=None*, *proxy=None*, *subdomains=None*)

Bases: *cesiumpy.provider.ImageryProvider*

url [str] The base URL for the WMTS GetTile operation (for KVP-encoded requests) or the tile-URL template (for RESTful requests). The tile-URL template should contain the following variables: {style}, {TileMatrixSet}, {TileMatrix}, {TileRow}, {TileCol}. The first two are optional if actual values are hardcoded or not required by the server. The {s} keyword may be used to specify subdomains.

layer [str] The layer name for WMTS requests.

style [str] The style name for WMTS requests.

format [str, default 'image/jpeg'] The MIME type for images to retrieve from the server.

tileMatrixSetID [str] The identifier of the TileMatrixSet to use for WMTS requests.

tileMatrixLabels [list] optional A list of identifiers in the TileMatrix to use for WMTS requests, one per TileMatrix level.

rectangle [Rectangle, default Rectangle.MAX_VALUE] The rectangle covered by the layer.

tilingScheme [TilingScheme, default new GeographicTilingScheme()] The tiling scheme corresponding to the organization of the tiles in the TileMatrixSet.

ellipsoid [Ellipsoid] The ellipsoid. If not specified, the WGS84 ellipsoid is used.

tileWidth [int, default 256] optional The tile width in pixels.

tileHeight [int, default 256] The tile height in pixels.

minimumLevel [int, default 0] The minimum level-of-detail supported by the imagery provider.

maximumLevel [int] The maximum level-of-detail supported by the imagery provider, or undefined if there is no limit.

credit [Credit or str] A credit for the data source, which is displayed on the canvas.

proxy [Proxy] A proxy to use for requests. This object is expected to have a getURL function which returns the proxied URL.

subdomains [str or list of str, default 'abc'] The subdomains to use for the {s} placeholder in the URL template. If this parameter is a single string, each character in the string is a subdomain. If it is an array, each element in the array is a subdomain.

format
A trait for unicode strings.

layer
A trait for unicode strings.

style
A trait for unicode strings.

tileMatrixSetID
A trait for unicode strings.

class cesiumpy.scene.**Scene**(*widget*)
Bases: cesiumpy.base._CesiumObject

primitives

script

widget
A trait whose value must be an instance of a specified class.

The value can also be an instance of a subclass of the specified class.

Subclasses can declare default classes by overriding the klass attribute

class cesiumpy.viewer.**Viewer**(*divid=None*, *width=u'100%'*, *height=u'100%'*, *animation=None*, *baseLayerPicker=None*, *fullscreenButton=None*, *geocoder=None*, *homeButton=None*, *infoBox=None*, *sceneModePicker=None*, *selectionIndicator=None*, *timeline=None*, *navigationHelpButton=None*, *navigationInstructionsInitiallyVisible=None*, *scene3DOnly=None*, *clock=None*, *selectedImageryProviderViewModel=None*, *imageryProviderViewModels=None*, *selectedTerrainProviderViewModel=None*, *terrainProviderViewModels=None*, *imageryProvider=None*, *terrainProvider=None*, *skyBox=None*, *skyAtmosphere=None*, *fullscreenElement=None*, *useDefaultRenderLoop=None*, *targetFrameRate=None*, *showRenderLoopErrors=None*, *automaticallyTrackDataSourceClocks=None*, *contextOptions=None*, *sceneMode=None*, *mapProjection=None*, *globe=None*, *orderIndependentTranslucency=None*, *creditContainer=None*, *dataSources=None*, *terrainExaggeration=None*)
Bases: cesiumpy.base._CesiumBase

divid [str] id string used in div tag

width [str] width of div tag, should be provided as css format like "100%" or "100px"

height [str] height of div tag, should be provided as css format like "100%" or "100px"

animation [bool, default True] If set to false, the Animation widget will not be created.

baseLayerPicker [bool, default True] If set to false, the BaseLayerPicker widget will not be created.

fullscreenButton [bool, default True] If set to false, the FullscreenButton widget will not be created.

geocoder [bool, default True] If set to false, the Geocoder widget will not be created.

homeButton [bool, default True] If set to false, the HomeButton widget will not be created.

infoBox [bool, default True] If set to false, the InfoBox widget will not be created.

sceneModePicker [bool, default True] If set to false, the SceneModePicker widget will not be created.

selectionIndicator [bool, default True] If set to false, the SelectionIndicator widget will not be created.

timeline [bool, default True] If set to false, the Timeline widget will not be created.

navigationHelpButton [bool, default True] If set to the false, the navigation help button will not be created.

navigationInstructionsInitiallyVisible [bool, defaut True] True if the navigation instructions should initially be visible, or false if the should not be shown until the user explicitly clicks the button.

scene3DOnly [bool, default False] When true, each geometry instance will only be rendered in 3D to save GPU memory.

clock [Clock, default new Clock()] The clock to use to control current time.

selectedImageryProviderViewModel [ProviderViewModel] The view model for the current base imagery layer, if not supplied the first available base layer is used. This value is only valid if options.baseLayerPicker is set to true.

imageryProviderViewModels [list of ProviderViewModel, default createDefaultImageryProviderViewModels()] The list of ProviderViewModels to be selectable from the BaseLayerPicker. This value is only valid if options.baseLayerPicker is set to true.

selectedTerrainProviderViewModel [ProviderViewModel] The view model for the current base terrain layer, if not supplied the first available base layer is used. This value is only valid if options.baseLayerPicker is set to true.

terrainProviderViewModels [list of ProviderViewModel, default createDefaultTerrainProviderViewModels()] The list of ProviderViewModels to be selectable from the BaseLayerPicker. This value is only valid if options.baseLayerPicker is set to true.

imageryProvider [ImageryProvider, default new BingMapsImageryProvider()] The imagery provider to use. This value is only valid if options.baseLayerPicker is set to false.

terrainProvider [TerrainProvider, default new EllipsoidTerrainProvider()] The terrain provider to use

skyBox [SkyBox] The skybox used to render the stars. When undefined, the default stars are used.

skyAtmosphere [SkyAtmosphere] Blue sky, and the glow around the Earth's limb. Set to false to turn it off.

fullscreenElement [Element or str, default document.body] The element or id to be placed into fullscreen mode when the full screen button is pressed.

useDefaultRenderLoop [bool, default True] True if this widget should control the render loop, false otherwise.

targetFrameRate [float] The target frame rate when using the default render loop.

showRenderLoopErrors [bool, default True] If true, this widget will automatically display an HTML panel to the user containing the error, if a render loop error occurs.

automaticallyTrackDataSourceClocks [bool, default True] If true, this widget will automatically track the clock settings of newly added DataSources, updating if the DataSource's clock changes. Set this to false if you want to configure the clock independently.

contextOptions [Object] Context and WebGL creation properties corresponding to options passed to Scene.

sceneMode [SceneMode, default SceneMode.SCENE3D] The initial scene mode.

mapProjection [MapProjection, default new GeographicProjection()] The map projection to use in 2D and Columbus View modes.

globe [Globe, default new Globe(mapProjection.ellipsoid)] The globe to use in the scene. If set to false, no globe will be added.

orderIndependentTranslucency [bool, default True] If true and the configuration supports it, use order independent translucency.

creditContainer [Element or str] The DOM element or ID that will contain the CreditDisplay. If not specified, the credits are added to the bottom of the widget itself.

dataSources [list of DataSource] The collection of data sources visualized by the widget. If this parameter is provided, the instance is assumed to be owned by the caller and will not be destroyed when the viewer is destroyed.

terrainExaggeration [float, default 1.] A scalar used to exaggerate the terrain. Note that terrain exaggeration will not modify any other primitive as they are positioned relative to the ellipsoid.

animation
A boolean (True, False) trait.

automaticallyTrackDataSourceClocks
A boolean (True, False) trait.

baseLayerPicker
A boolean (True, False) trait.

fullscreenButton
A boolean (True, False) trait.

geocoder
A boolean (True, False) trait.

homeButton
A boolean (True, False) trait.

infoBox
A boolean (True, False) trait.

navigationHelpButton
A boolean (True, False) trait.

navigationInstructionsInitiallyVisible
A boolean (True, False) trait.

plot

sceneModePicker
A boolean (True, False) trait.

selectionIndicator
A boolean (True, False) trait.

timeline
A boolean (True, False) trait.

class cesiumpy.widget.**CesiumWidget**(*divid=None*, *width=u'100%'*, *height=u'100%'*, *clock=None*, *imageryProvider=None*, *terrainProvider=None*, *skyBox=None*, *skyAtmosphere=None*, *sceneMode=None*, *scene3DOnly=None*, *orderIndependentTranslucency=None*, *mapProjection=None*, *globe=None*, *useDefaultRenderLoop=None*, *targetFrameRate=None*, *showRenderLoopErrors=None*, *contextOptions=None*, *creditContainer=None*, *terrainExaggeration=None*)

Bases: cesiumpy.base._CesiumBase

divid [str] id string used in div tag

width [str] width of div tag, should be provided as css format like "100%" or "100px"

height [str] height of div tag, should be provided as css format like “100%” or “100px”

clock [Clock, default new Clock()] The clock to use to control current time.

imageryProvider [ImageryProvider, default new BingMapsImageryProvider()] The imagery provider to serve as the base layer. If set to false, no imagery provider will be added.

terrainProvider [TerrainProvider, default new EllipsoidTerrainProvider()] The terrain provider.

skyBox [SkyBox] The skybox used to render the stars. When undefined, the default stars are used. If set to false, no skyBox, Sun, or Moon will be added.

skyAtmosphere [SkyAtmosphere] Blue sky, and the glow around the Earth’s limb. Set to false to turn it off.

sceneMode [SceneMode, default SceneMode.SCENE3D] The initial scene mode.

scene3DOnly [bool, default False] When true, each geometry instance will only be rendered in 3D to save GPU memory.

orderIndependentTranslucency [bool, default True] If true and the configuration supports it, use order independent translucency.

mapProjection [MapProjection, default new GeographicProjection()] The map projection to use in 2D and Columbus View modesself.

globe [Globe, default new Globe(mapProjection.ellipsoid)] The globe to use in the scene. If set to false, no globe will be added.

useDefaultRenderLoop [bool, default True] True if this widget should control the render loop, false otherwise.

targetFrameRate [int] The target frame rate when using the default render loop.

showRenderLoopErrors [bool, default True] If true, this widget will automatically display an HTML panel to the user containing the error, if a render loop error occurs.

contextOptions [Object] Context and WebGL creation properties corresponding to options passed to Scene.

creditContainer [Element or str] The DOM element or ID that will contain the CreditDisplay. If not specified, the credits are added to the bottom of the widget itself.

terrainExaggeration [float, default 1.] A scalar used to exaggerate the terrain. Note that terrain exaggeration will not modify any other primitive as they are positioned relative to the ellipsoid.

6.3 Module contents

C

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z