Cesium（セシウム）を試用するための

研修マニュアル【応用編】

Ver1.0

目次

１．はじめに

１－１．概要

本資料は、Cesium の試用について記した実習テキスト（基礎編）を習得した人向けに、基礎編で習得した事項を発展させるための技術テクニックを紹介する資料です。

資料作成にあたっては下記サイトを参考にし、また、それ以外の技術テクニックを紹介しています。

・GIS 実習オープン教材：Cesium 入門
https://gis-oer.github.io/gitbook/book/materials/web_gis/Cesium/Cesium.html

本資料で紹介する技術テクニックを応用することで、Cesium で表現される内容に追加情報を付与することができます。用途に応じて追加情報を付与し、より分かりやすい内容として Cesium を利用した情報提供を行うことができるようになります。

[用語解説]

本資料では聞きなれない用語が登場します。特に html ファイルを扱う際に使用するプログラム言語に関する専門用語が登場しますので、代表的な用語を下記に解説します。

＊	用語		説明
1	メソッド	method	オブジェクト指向言語（JavaScriptを含む）における用語。 オブジェクトに属する処理や操作のこと。
2	メンバー	member	オブジェクト指向プログラミング言語における、クラスまたはオブジェクトに所属するサブルーチンのこと。 メンバー関数とも言う。

1－2．紹介する技術テクニック

本資料で紹介する技術テクニックは下表の通りです。

	章	技術テクニック	説明
1	2-1	機能のオン／オフ	Cesiumに標準搭載されている基本機能のオン／オフの切り替え方法について説明します。
2	2-2	レイヤの追加	空間座標を利用し、ポイント、ライン、ポリゴンの各追加方法について説明します。
3	2-3	視点の変更	Cesiumで閲覧する際の初期視点の設定方法について説明します。
4	3-1	ラベルの追加	空間座標を利用し、ラベルの追加方法について説明します。
5	3-2	ベースマップの 地形の変更	Cesiumで閲覧する際に読み込むベースマップの地形の変更方法について説明します。
6	3-3	ホームボタンの 設定変更	Cesiumに標準搭載されているホームボタンで設定される空間情報の変更方法について説明します。
7	3-4	物件の情報による 色の変更	物件が所有している情報にあわせた色の変更方法について説明します。

は前編として『GIS 実習オープン教材：Cesium 入門』で紹介されている項目の具体的な手法を説明しています。

GIS 実習オープン教材：Cesium 入門

https://gis-oer.github.io/gitbook/book/materials/web_gis/Cesium/Cesium.html

は後編として前編で紹介されていないプラスアルファの技術テクニックを説明しています。

本資料で紹介する技術テクニック以外にも多種多様な技術テクニックが存在します。
詳しくは Cesium 公式のドキュメントを参照してください。

・Cesium 公式ドキュメント（英語）
https://cesium.com/learn/cesiumjs/ref-doc/index.html

１－３．本研修で使用するデータ

本研修で使用するデータは Cesium の試用について記した実習テキスト（基礎編）の２－４で導入した Cesium のデータおよび、同２－６で準備した研修用のデータを用います。

また、同２－４で導入した Cesium のデータ内にある「HelloWorld.html」（以下、HelloWorld.html ファイルとします）の内容を編集しながら技術テクニックを説明していきます。

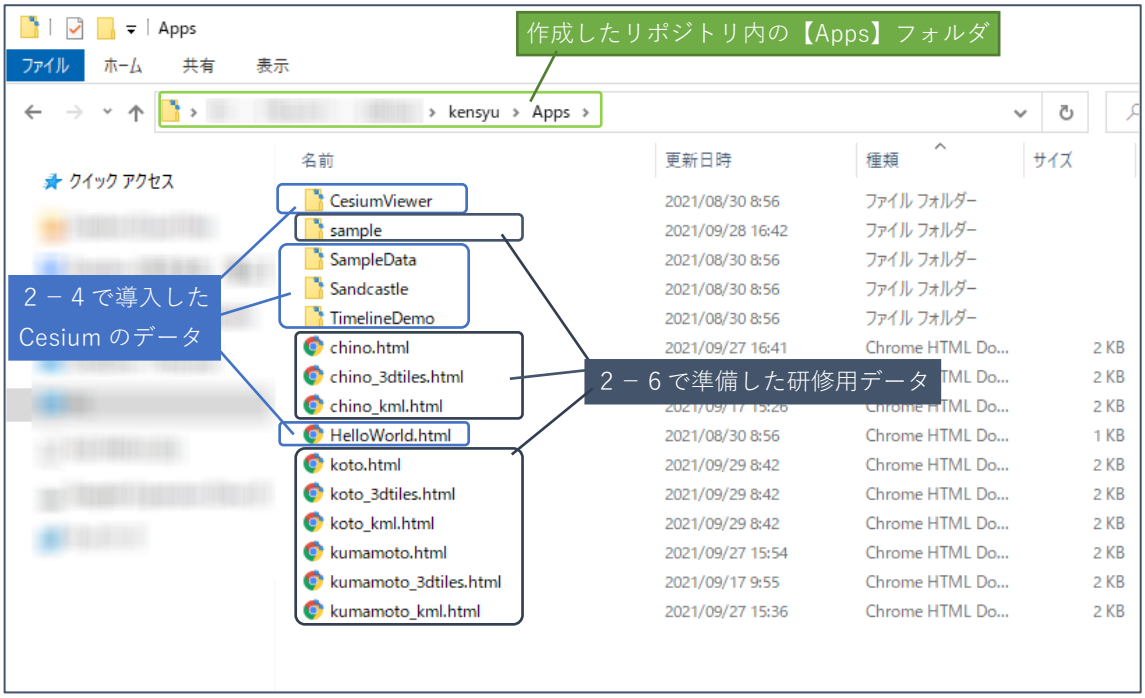


図１－３．使用するデータの内容

また、各々で実際に HelloWorld.html ファイルを編集し習得する人向けに、本研修で紹介する技術テクニックの記述例の html ファイルを用意しています。

R3 年度 Web 研修 ポータルサイトの「2-1-3.CESIUM の試用」ページにアクセスしダウンロードしてください。

１－４．使用言語

本研修で使用する HelloWorld.html ファイルは主に「HTML」言語と「JavaScript」言語で構成されています。

従って、HelloWorld.html ファイルを編集し、技術テクニックを説明するにあたっても「HTML」言語と「JavaScript」言語を使用しています。また、HelloWorld.html ファイルへの記述ルールは「HTML」言語と「JavaScript」言語に準拠しています。

なお、「HTML」言語と「JavaScript」言語の詳しい説明については本研修では紹介しません。

１－５．コード解説の補足資料

本資料各章においてコードを解説していますが、本研修にあわせた簡易的な内容となっています。

各章のコード解説におけるそれぞれのメソッドやメンバーの詳しい説明は Cesium 公式のドキュメントを参照してください。

・Cesium 公式ドキュメント（英語）
https://cesium.com/learn/cesiumjs/ref-doc/index.html

１－６．注意事項

実習テキスト（基礎編）と同様に、本研修でも Web ブラウザ上で Cesium を試用するために、必要なデータをインターネット上にアップロードします。

アップロードされたデータは**誰でも閲覧やダウンロードができる**状態となりますので、各々で準備するデータを用いて Cesium を試用する場合は**非公開データを避け**、一般公開されているデータを用いて試用してください。

2．技術テクニック：前編

ここでは、『GIS 実習オープン教材：Cesium 入門』で紹介されている以下の項目について、その具体的な手法について説明します。

	章	技術テクニック	説明
1	2-1	機能のオン／オフ	Cesiumに標準搭載されている基本機能のオン／オフの切り替え方法について説明します。
2	2-2	レイアの追加	空間座標を利用し、ポイント、ライン、ポリゴンの各追加方法について説明します。
3	2-3	視点の変更	Cesiumで閲覧する際の初期視点の設定方法について説明します。

具体的な手法の説明に際しては、HelloWorld.html ファイルをメモ帳などのテキストエディタで開き、内容を編集しながら説明していきます。

HelloWorld.html ファイルをメモ帳で開くと、下図の内容が見れます。この内容を編集していきます。

図２．HelloWorld.html の内容

２－１．機能のオン／オフ

[説明]

ここでは、Cesium の画面で表示される基本機能のオン／オフについて説明します。

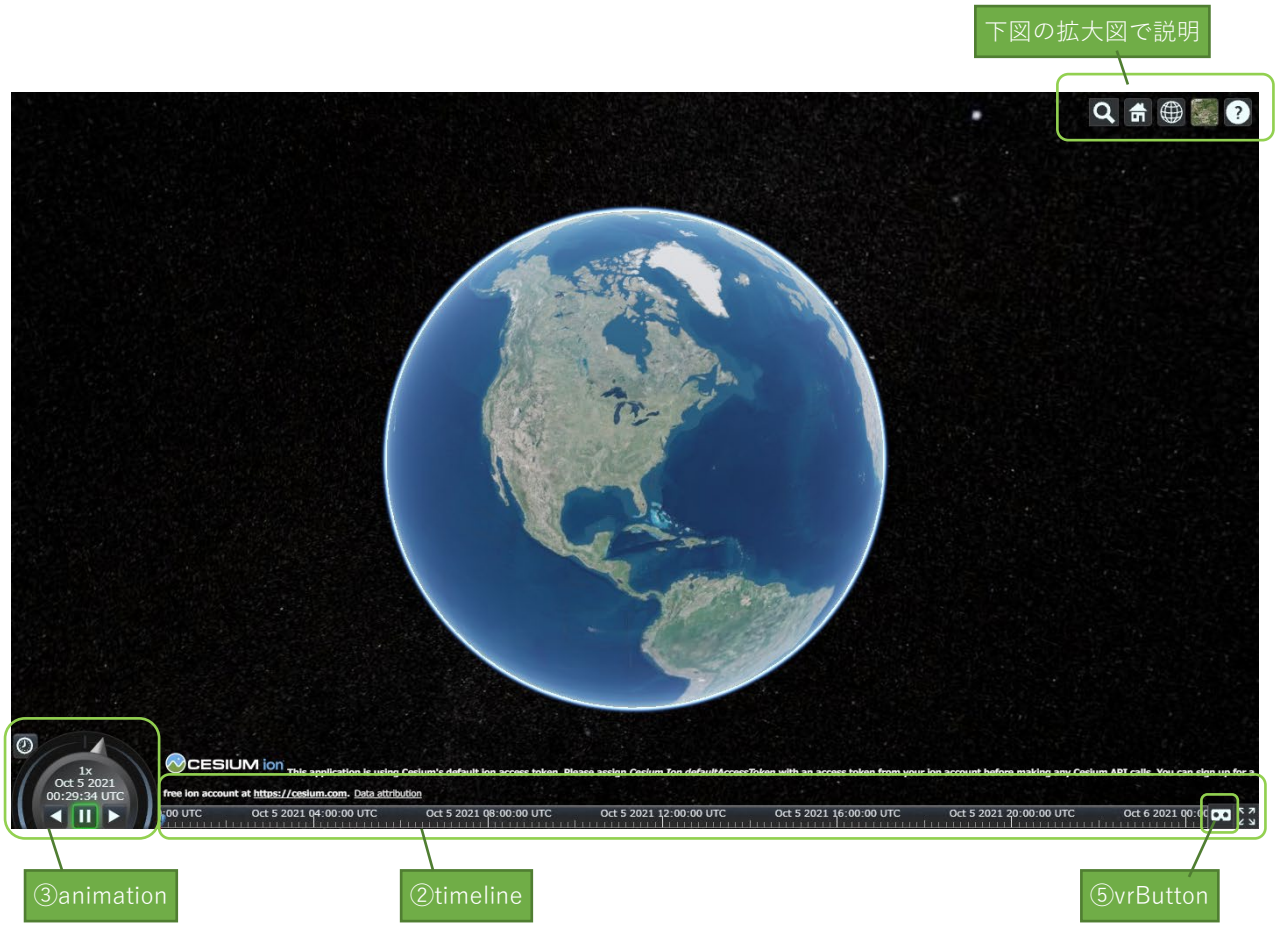
Cesium の画面で表示される基本機能は、以下のように「new Cesium.Viewer」で宣言されるメソッドにおいて、各メンバーの値を“true”または“false”で設定することでオン／オフの管理をすることができます。

```
var viewer = new Cesium.Viewer(”cesiumContainer”, {
  baseLayerPicker: true,
  timeline : false,
  animation : false,
  homeButton: false,
  vrButton: true,
  geocoder:false,
  sceneModePicker:false,
  navigationHelpButton:false
});
```

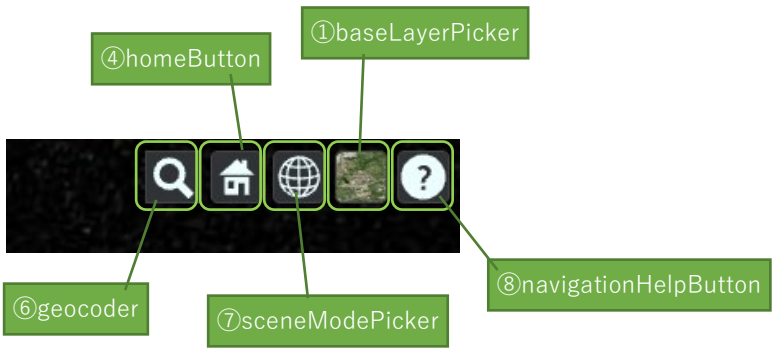
各メンバーの説明は下表の通りです。

No	メンバー	説明
①	baseLayerPicker	ベースマップを選択するメニューの表示のオン／オフを設定します。
②	timeline	タイムラインの表示のオン／オフを設定します。
③	animation	アニメーションの操作メニューの表示のオン／オフを設定します。
④	homeButton	ホームボタンの表示のオン／オフを設定します。
⑤	vrButton	VRボタンの表示のオン／オフを設定します。
⑥	geocoder	住所および主要ランドマーク等の検索ボタンの表示のオン／オフを設定します。
⑦	sceneModePicker	地図の投影方法を選択するメニューの表示のオン／オフを設定します。
⑧	navigationHelpButton	画面の操作方法のヘルプメニューの表示のオン／オフを設定します。

全てのメンバーの設定値を“true”にした場合、下図のように表示されます。



【拡大図】



なお、HelloWorld.html ファイルを開くと、31 行目に下記のように表記されています。

```
var viewer = new Cesium.Viewer(”cesiumContainer”);
```

これはデフォルトの表記となります。

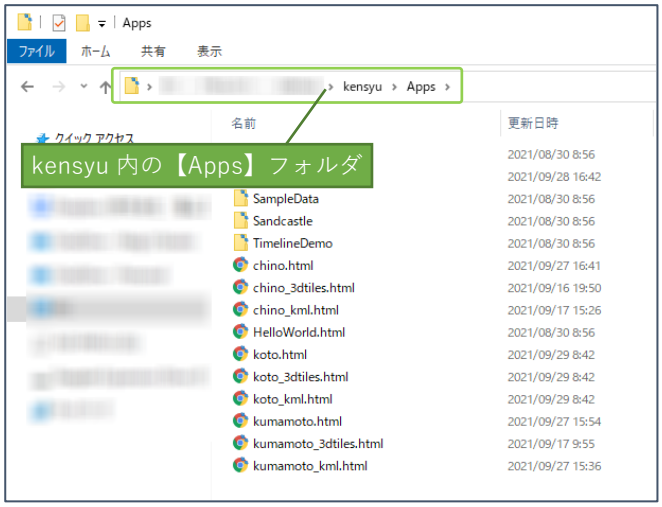
また、各メンバーの設定値を下記のように設定することで、デフォルトと同義になります。
（vrButton の設定値だけを false にし、それ以外の設定値は true で設定したものです。）

```
var viewer = new Cesium.Viewer(”cesiumContainer”, {
  baseLayerPicker: true,
  timeline : true,
  animation : true,
  homeButton: true,
  vrButton: fales,
  geocoder: true,
  sceneModePicker: true,
  navigationHelpButton: true
});
```

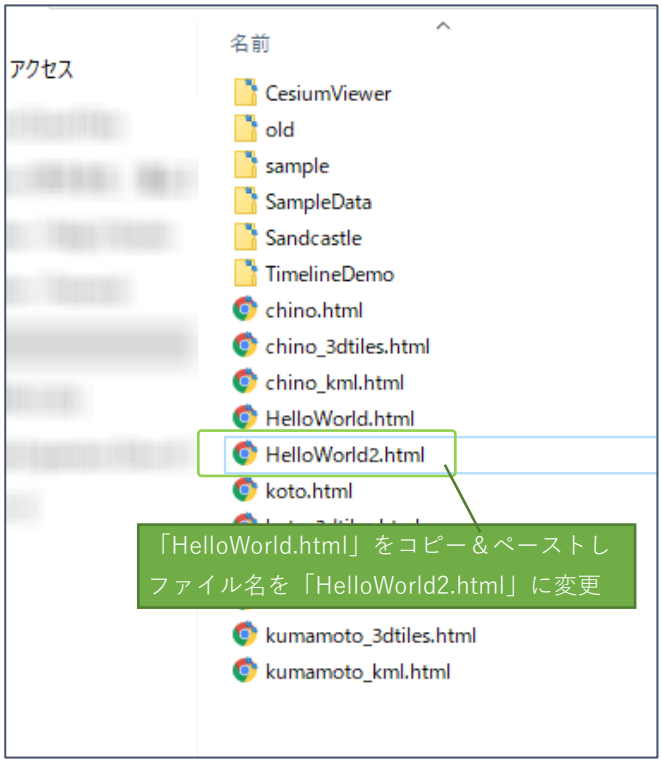
[実践]

実際に HelloWorld.html ファイルをコピーし、その内容を編集して表示内容の変更を試します。

① 実習テキスト（基礎編）の２－６③の手順に習い、リポジトリ（kensyu）内の【Apps】フォルダを開きます。



② 「HelloWorld.html」をコピー＆ペーストし、ファイル名を「HelloWorld2.html」に変更します。

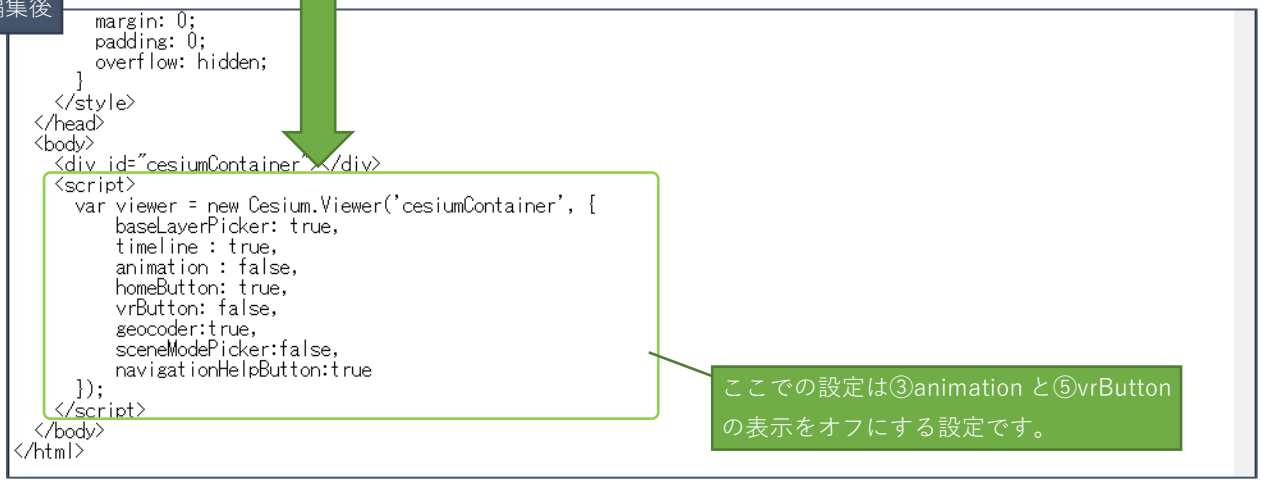


③ 「HelloWorld2.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld2.html 編集前

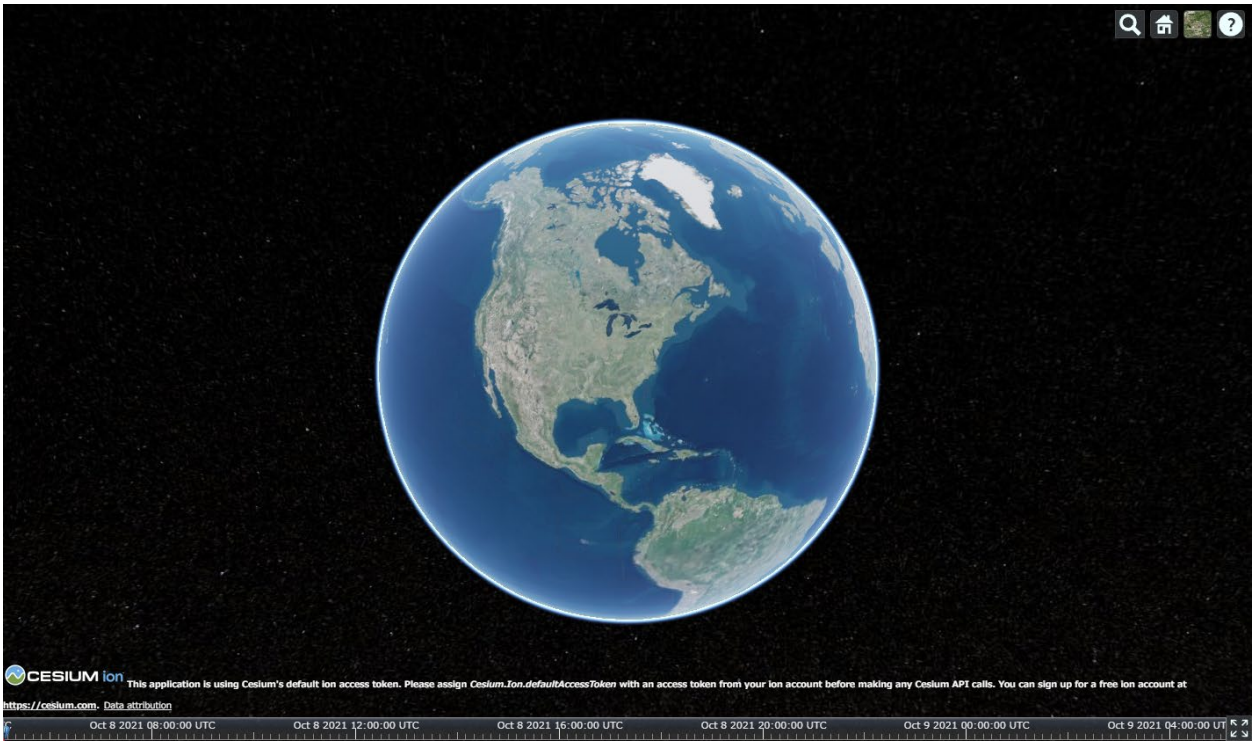
編集後



⑤ 実習テキスト（基礎編）の２－７に従って、GitHub Desktop 経由で「HelloWorld2.html」をアップロードします。

⑥ 実習テキスト（基礎編）の２－８を参考に、アップロードした「HelloWorld2.html」を表示させます。

表示させた結果は下図のようになります。



p9 の全てのメンバーを True にした場合の画面構成と比べ、以下の３点が非表示になっています。

（１）左下にあったアニメーションの操作メニュー（③animation）

（２）右下にあった VR ボタンの（⑤vrButton）

（３）右上にあった地図の投影方法を選択するメニュー（⑦sceneModePicker）

興味のある方はその他の設定値を変更し、表示がどのように変わるか試してみてください。

以上で本項目は終了です。

２－２．レイヤの追加

[説明]

ここでは、Cesium 上に空間座標をもつデータを追加する方法について説明します。

空間座標をもつデータとは「緯度」「経度」「高さ（標高)」をもつデータであり、この空間座標を用いて Cesium 上に「ポイント」「ライン」「ポリゴン」として追加することができます。

以降に説明する各実践でも、本書２－１と同様に HelloWorld.html ファイルをコピーし、その内容を編集して実践していきます。

[実践]　ポリゴンの追加

ここでは Cesium 上に「ポリゴン」の追加を試します。

例題として、東京都庁の位置に四角形のポリゴンを追加することを実践します。

この章で挙げる html コードの記述例は「HelloWorld3_Example.html」を参考にしてください。

「ポリゴン」は指定された空間座標の「緯度」と「経度」で示される点を角にして、それぞれの角を結ぶことで形成されます。三角形の「ポリゴン」の場合は３点、四角形の場合は４つの点、五角形の場合は５つの点…、のように形状に合わせて空間座標の点の設定が必要になります。

例題の東京都庁の位置に四角形のポリゴンを配置することに対しても、４つの点（以下、ポイント群）の空間座標の「緯度」と「経度」を予め取得する必要があります。巻末の『【補足資料２】空間座標の「緯度」と「経度」の求め方』を参考に、これらの「緯度」と「経度」を取得してください。

---

① 巻末の『【補足資料２】空間座標の「緯度」と「経度」の求め方』を参考に、東京都庁に配置するポイント群の「緯度」と「経度」を求めます。

---

② 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

③ 「HelloWorld.html」をコピー＆ペーストし、ファイル名を「HelloWorld3.html」に変更します。

---

④ 「HelloWorld3.html」をメモ帳で開きます。

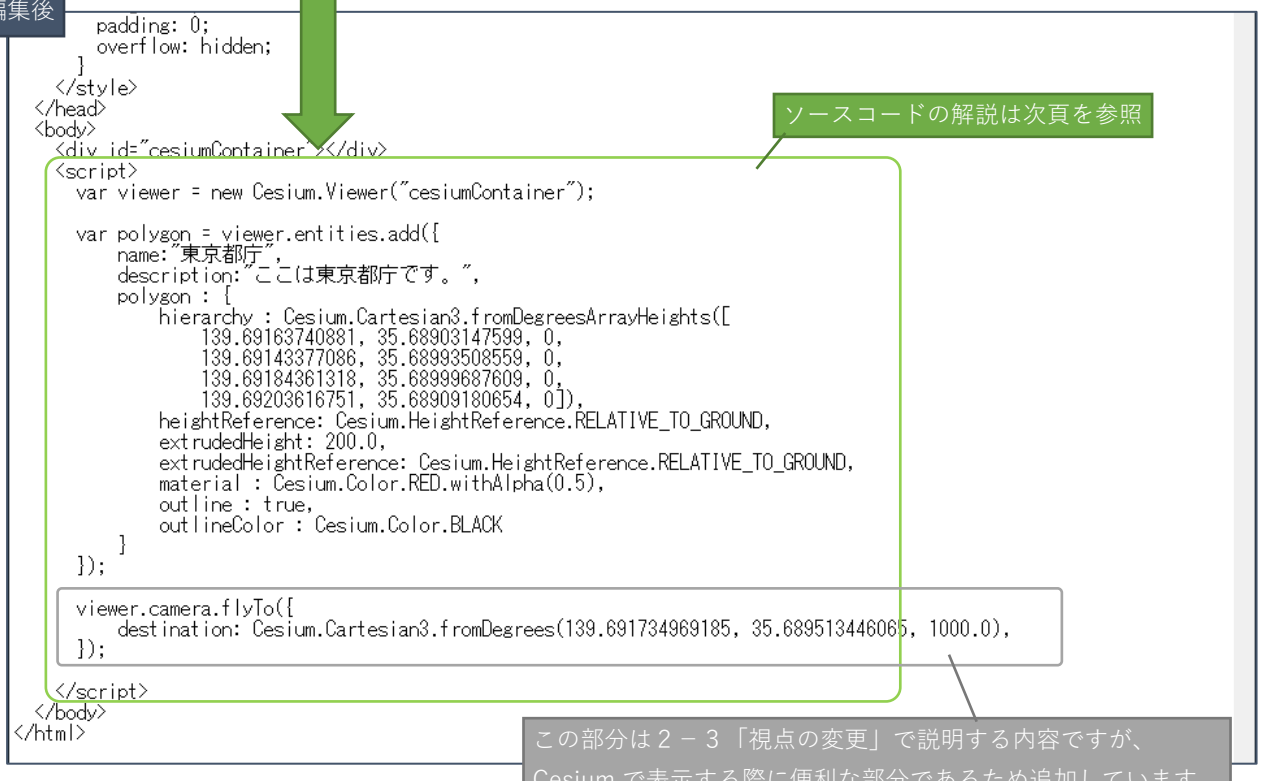
⑤ 下図に従って、内容を編集し、上書き保存します。

HelloWorld3.html 編集前

```
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Use correct character set. -->
    <meta charset="utf-8" />
    <!-- Tell IE to use the latest, best version. -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <!-- Make the application on mobile take up the full browser screen and disable user scaling. -->
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no"
    />
    <title>Hello World!</title>
    <script src="../Build/Cesium/Cesium.js"></script>
    <style>
      @import url(../Build/Cesium/Widgets/widgets.css);
      html,
      body,
      #cesiumContainer {
        width: 100%;
        height: 100%;
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");
    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後

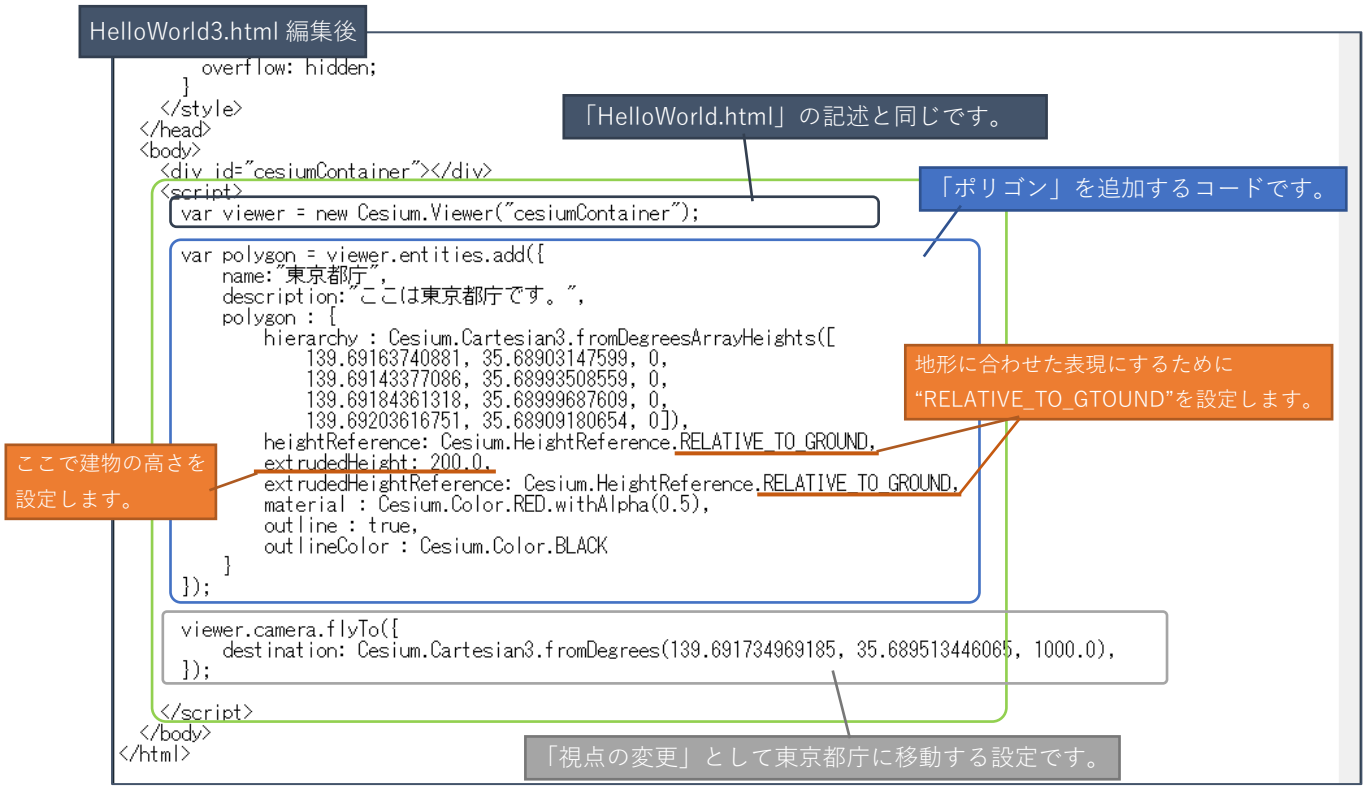


```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var polygon = viewer.entities.add({
          name:"東京都庁",
          description:"ここは東京都庁です。",
          polygon : {
              hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                  139.69163740881, 35.68903147599, 0,
                  139.69143377086, 35.68993508559, 0,
                  139.69184361318, 35.68999687609, 0,
                  139.69203616751, 35.68909180654, 0]),
              heightReference: Cesium.HeightReference.RELATIVE_TO_GROUND,
              extrudedHeight: 200.0,
              extrudedHeightReference: Cesium.HeightReference.RELATIVE_TO_GROUND,
              material : Cesium.Color.RED.withAlpha(0.5),
              outline : true,
              outlineColor : Cesium.Color.BLACK
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.691734969185, 35.689513446065, 1000.0),
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

この部分は２－３「視点の変更」で説明する内容ですが、
Cesium で表示する際に便利な部分であるため追加しています。

【編集部分の解説】



【ポリゴンを追加する部分のコード解説】

ポリゴンを追加するためには「veiwer.entities.add」メソッドを用います。

「veiwer.entities.add」で追加する際、表示する名称や説明文、ポリゴンの形状などを設定します。

ポリゴンの形状の設定は青枠内の 4 行目にある「polygon」内で行っており、ポリゴンを配置するポイント群の空間座標や色、高さ、辺の色などをメンバーの値に設定しています。各メンバーの解説については次頁の表の通りです。

また、本章冒頭でも示した通り、ポリゴンを配置するポイント群の空間座標の「緯度」と「経度」の求め方については巻末の『【補足資料２】空間座標の「緯度」と「経度」の求め方』を参照してください。

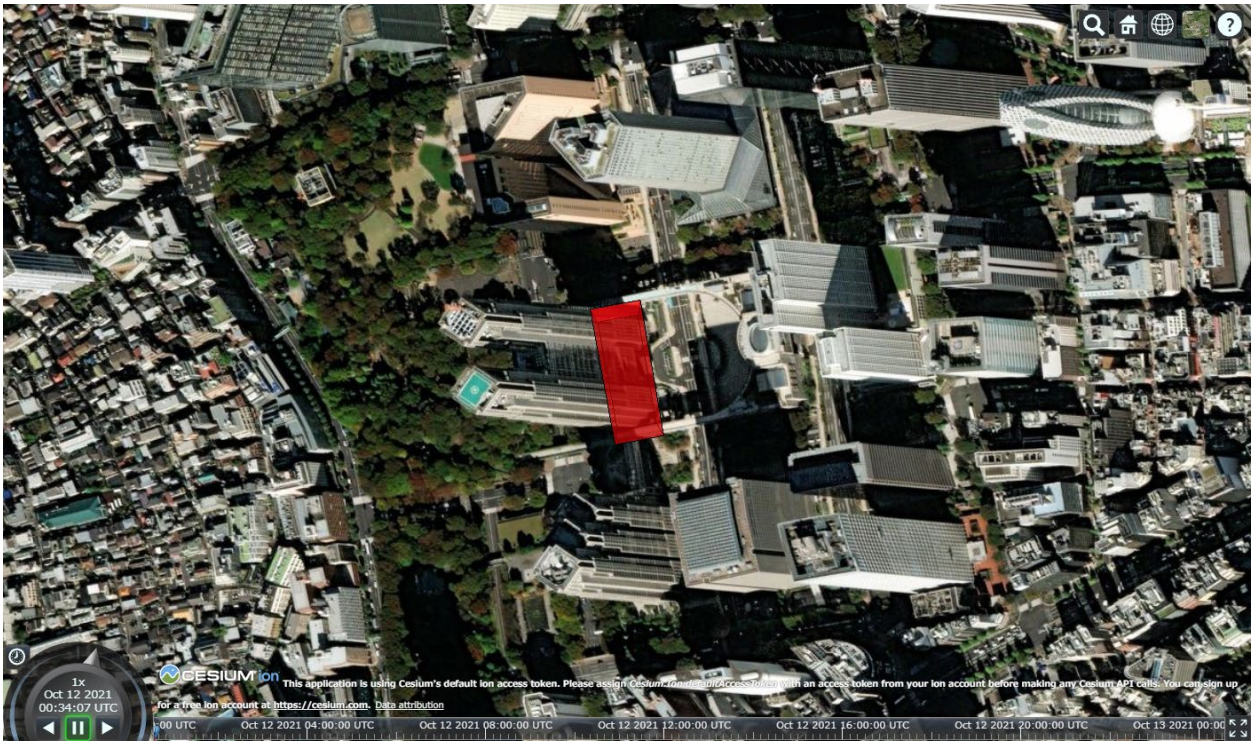
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.entities.add()・・・オブジェクトを追加するメソッドの呼び出し部。
2	name:	Cesium 上に表示されたポリゴンをクリックした際に表示される情報窓のタイトルの設定部分。 例題では“東京都庁”と設定。
3	description:	Cesium 上に表示されたポリゴンをクリックした際に表示される情報に記載する内容を設定。 例題では"ここは東京都庁です。"と設定。
4	polygon:	「ポリゴン」を示すポイント群の配置や太さ、色などの設定部分。
5～9	hierarchy:	「ポリゴン」を配置するポイント群の空間座標の設定部分。 Cesium.Cartesian3.fromDegreesArrayHeights()メソッドを利用してポイントそれぞれの緯度、経度、ポイントを置く高さを指定している。 メソッドには緯度、経度、ポイントを置く高さの順にカンマ区切りで記述し、記述されたポイントの順番にポリゴンが形成される。 例題では４ヵ所のポイントをつなげ、“東京都庁”を示している。ポイントを置く位置の高さはいずれも 0 で設定されている。 「ポリゴン」を配置するポイント群の空間座標のうち「緯度」と「経度」の求め方については【補足資料２】を参照。
10	heightReference:	「ポリゴン」を置く位置の設定部分。 地形に合わせて表現するためには「RELATIVE_TO_GROUND」を設定する。
11	extrudedHeight:	「ポリゴン」を押し出す幅（高さ）の設定部分。 例題では 200m を押し出す設定。
12	extrudedHeightReference:	「ポリゴン」を押し出す基準地点の設定部分。 地形に合わせて表現するためには「RELATIVE_TO_GROUND」を設定する。
13	material:	配置する「ポリゴン」本体の色の設定部分。 例題ではポリゴンの色を赤で設定し、withAlpha()部分で透過率＝50%を設定している。透過させない場合は withAlpha(1)と記述する。 例題以外の色を設定する場合は巻末の『【補足資料３】色の設定部分の記述方法』に示す表の通り。
14	outline:	配置する「ポリゴン」の辺の着色の有無の設定部分。true＝着色する、false＝着色しない。
15	outlineColor:	配置する「ポリゴン」の辺の色の設定部分。例題では黒で設定。 また、ここでは透過させない設定として“Cesium.Color.BLACK”と記述しているが、透過させたい場合は上記の“material:”での記載と同様に withAlpha()を用いて“Cesium.Color.BLACK.withAlpha(0.5)”のように記述する。 こちらも、例題以外の色を設定する場合は巻末の『【補足資料３】色の設定部分の記述方法』に示す表の通り。

⑥ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld3.html」をアップロードします。

⑦ 本書２－１⑥と同様に、アップロードした「HelloWorld3.html」を表示させます。

表示させた結果は下図のようになります。



ポリゴンをクリックすると右上に情報が表示されます。

【参考：複数のポリゴンを追加したい場合】

複数のポイントを追加したい場合、複数のポイントや複数のラインの追加と同様に、一つずつ追加します。また、“hierarchy”で設定するポイントの空間座標を細かく設定することで、複雑な形状のポリゴンを追加することができます。

複数ポリゴンの追加例

```
<div id="cesiumContainer"></div>
<script>
  var viewer = new Cesium.Viewer("cesiumContainer");

  var polygon = viewer.entities.add({
      name:"国会議事堂",
      description:"ここは国家議事堂です。",
      polygon : {
          hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
              139.74461658300, 35.67502420610, 0,
              139.74455471700, 35.67525263430, 0,
              139.74463324000, 35.67527404950, 0,
              139.74453092300, 35.67567141940, 0,
              139.74444764200, 35.67565238380, 0,
              139.74436436[illegible]595457530, 0,
              139.7444[illegible]036990, 0,
              [illegible]
              139.74460932000, 35.67497899000, 0,
              139.74473317700, 35.67496234010, 0,
              139.74470938200, 35.67504086230, 0]),
          extrudedHeight: 100.0,
          width : 5,
          material : Cesium.Color.fromBytes(255,0,255,128),
          outline : true,
          outlineColor : Cesium.Color.BLACK
      }
  });

  var polygon = viewer.entities.add({
      name:"新国立競技場",
      description:"ここは新国立競技場です。",
      polygon : {
          hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
              139.71342970900, 35.67864098400, 0,
              139.71365571500, 35.67886397910, 0,
              139.71391206100, 35.67904479930, 0,
              139.71418889600, 35.67917649550, 0,
              139.7144755[illegible]5400700, 0,
              [illegible]
              139.71301281300, 35.67781072250, 0,
              139.71310196100, 35.67810403840, 0,
              139.71324273000, 35.67838438340, 0]),
          extrudedHeight: 100.0,
          width : 5,
          material : Cesium.Color.AQUA.withAlpha(0.5),
          outline : true,
          outlineColor : Cesium.Color.BLACK
      }
  });

    viewer.camera.flyTo({
        destination: Cesium.Cartesian3.fromDegrees(139.744, 35.662, 2000.0),
        orientation : {
            heading : Cesium.Math.toRadians(-30.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
        }
    });

</script>
</body>
</html>
```

「国会議事堂」のポリゴンを追加するコード

色の指定方法を Cesium.Color.fromBytes(red, green, blue, alpha) の記述方法で設定したもの。記述の説明については『【補足資料３】色の設定部分の記述方法』を参照。

「国立競技場」のポリゴンを追加するコード

表示させた結果は次頁のようになります。

複数ポリゴンを追加した html コードの記述例は「HelloWorld3_Example_EX.html」です。参考にしてください。

また、次頁以降で説明する「ポイント」および「ライン」を同一の画面上に表示させることも可能です。興味のある方は試してみてください。

以上で本項目は終了です。

[参考１] ポイントの追加

Cesium 上に「ポイント」の追加について説明します。
例題として、新国立競技場にポイントを追加することを実践します。
この章で挙げる html コードの記述例は「HelloWorld4_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld.html」をコピー＆ペーストし、ファイル名を「HelloWorld4.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld4.html」をメモ帳で開きます。

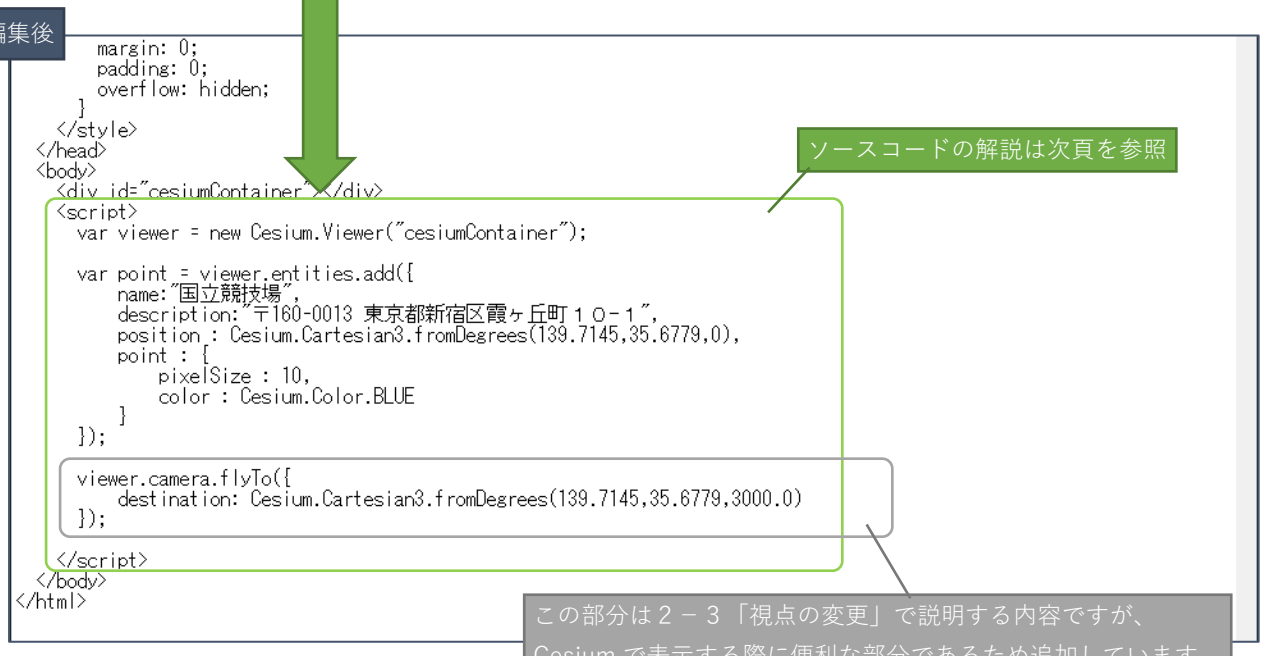
④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld4.html 編集前

```
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Use correct character set. -->
    <meta charset="utf-8" />
    <!-- Tell IE to use the latest, best version. -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <!-- Make the application on mobile take up the full browser screen and disable user scaling. -->
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no"
    />
    <title>Hello World!</title>
    <script src="../Build/Cesium/Cesium.js"></script>
    <style>
      @import url(../Build/Cesium/Widgets/widgets.css);
      html,
      body,
      #cesiumContainer {
        width: 100%;
        height: 100%;
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");
    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後



```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var point = viewer.entities.add({
          name:"国立競技場",
          description:"〒160-0013 東京都新宿区霞ヶ丘町１０－１",
          position : Cesium.Cartesian3.fromDegrees(139.7145,35.6779,0),
          point : {
              pixelSize : 10,
              color : Cesium.Color.BLUE
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.7145,35.6779,3000.0)
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

この部分は２－３「視点の変更」で説明する内容ですが、
Cesium で表示する際に便利な部分であるため追加しています。

【編集部分の解説】

HelloWorld4.html 編集後

```
        padding: 0;
        overflow: hidden;
    }
  </style>
</head>
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer");

    var point = viewer.entities.add({
        name:"国立競技場",
        description: '〒160-0013 東京都新宿区霞ヶ丘町１０－１',
        position : Cesium.Cartesian3.fromDegrees(139.7145,35.6779,0),
        point : {
            pixelSize : 10,
            color : Cesium.Color.BLUE
        }
    });

    viewer.camera.flyTo({
        destination: Cesium.Cartesian3.fromDegrees(139.7145,35.6779,3000.0)
    });

  </script>
</body>
</html>
```

「HelloWorld.html」の記述と同じです。

「ポイント」を追加するコードです。

「視点の変更」として新国立競技場に移動する設定です。

【ポイントを追加する部分のコード解説】

ポイントを追加するためにはポリゴンの追加と同様に「veiwer.entities.add」メソッドを用います。

「veiwer.entities.add」で追加する際、表示する名称や説明文および、ポイントの空間座標やポイントの大きさ、色などをメンバーの値に設定することで、用途に合ったポイントとして追加することができます。各メンバーの解説については下表の通りです。

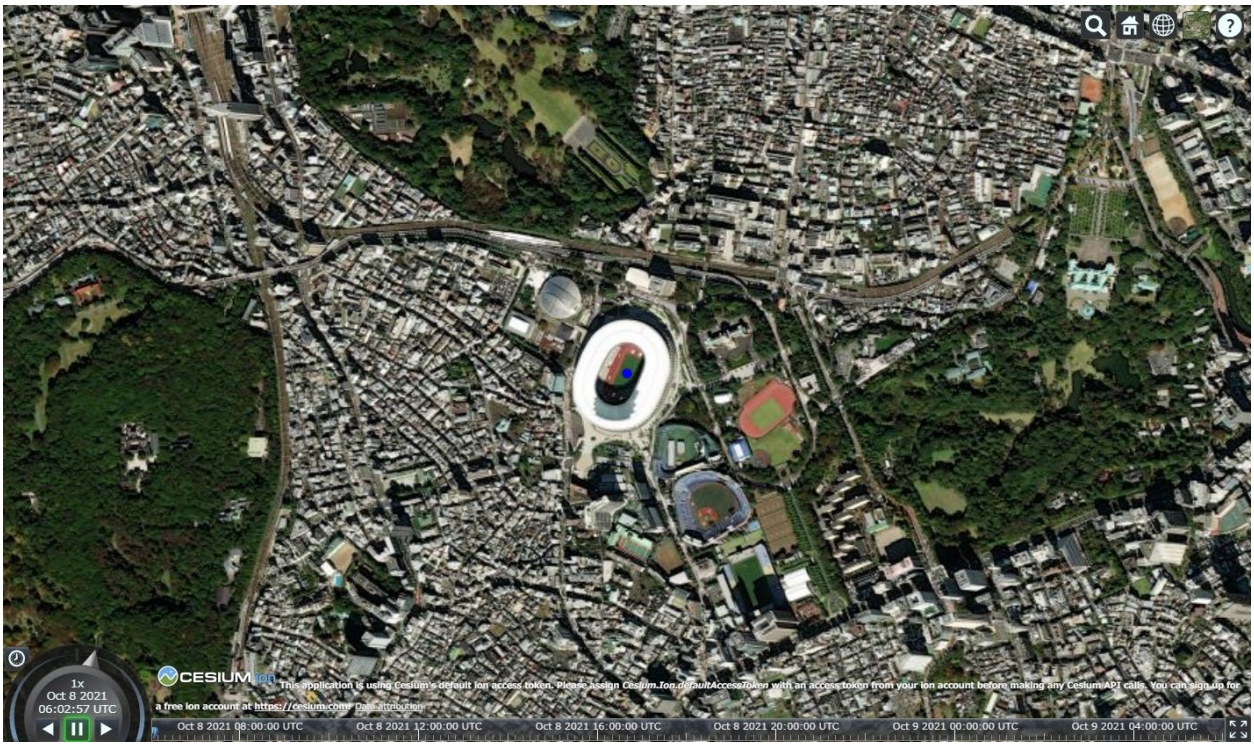
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.entities.add()・・・オブジェクトを追加するメソッドの呼び出し部。
2	name:	Cesium 上に表示されたポイントをクリックした際に表示される情報窓のタイトルの設定部分。 例題では“国立競技場”と設定。
3	description:	Cesium 上に表示されたポイントをクリックした際に表示される情報に記載する内容を設定。 例題では"〒160-0013 東京都新宿区霞ヶ丘町１０－１"と設定。
4	position:	「ポイント」を配置する空間座標の設定部分。Cesium.Cartesian3.fromDegrees()メソッドを利用して、緯度、経度、ポイントを置く高さを指定している。 メソッドには緯度、経度、ポイントを置く高さの順にカンマ区切りで記述する。 例題では“国立競技場”を示す緯度・経度を設定し、高さ＝0 で設定。
5	point:	配置する「ポイント」の大きさや色などの設定部分。
6	pixelSize:	配置する「ポイント」の大きさの設定部分。例題では 10 ピクセルで設定。
7	color:	配置する「ポイント」の色の設定部分。例題では青で設定。

⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld4.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld4.html」を表示させます。

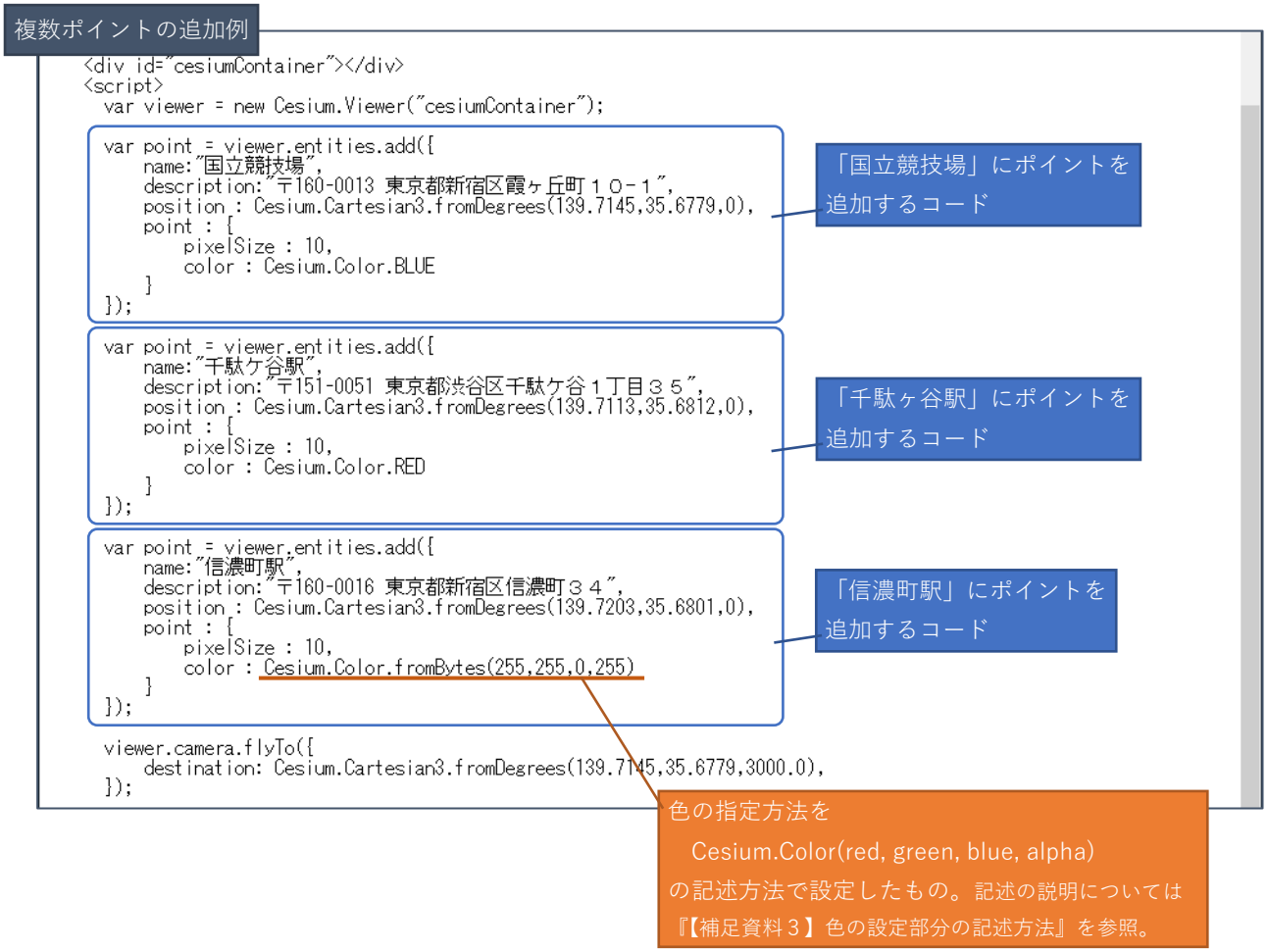
表示させた結果は下図のようになります。



ポイントをクリックすると右上に情報が表示されます。

【参考：複数のポイントを追加したい場合】

複数のポイントを追加したい場合、下図のようにポイントを一つずつ追加します。



表示させた結果は次頁のようになります。

複数ポイントを追加した html コードの記述例は「HelloWorld4_Example_EX.html」です。参考にしてください。

また、興味のある方はその他の施設のポイントの追加や、ポイントの色、大きさなどを変更し、表示がどのように変わるか試してみてください。

以上で本項目は終了です。

[参考２]　ラインの追加

Cesium 上に「ライン」の追加について説明します。
例題として、東京都で流れている荒川の一部にラインを追加することを実践します。
この章で挙げる html コードの記述例は「HelloWorld5_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld.html」をコピー＆ペーストし、ファイル名を「HelloWorld5.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld5.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld5.html 編集前

```
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Use correct character set. -->
    <meta charset="utf-8" />
    <!-- Tell IE to use the latest, best version. -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <!-- Make the application on mobile take up the full browser screen and disable user scaling. -->
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no"
    />
    <title>Hello World!</title>
    <script src="../Build/Cesium/Cesium.js"></script>
    <style>
      @import url(../Build/Cesium/Widgets/widgets.css);
      html,
      body,
      #cesiumContainer {
        width: 100%;
        height: 100%;
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");
    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後

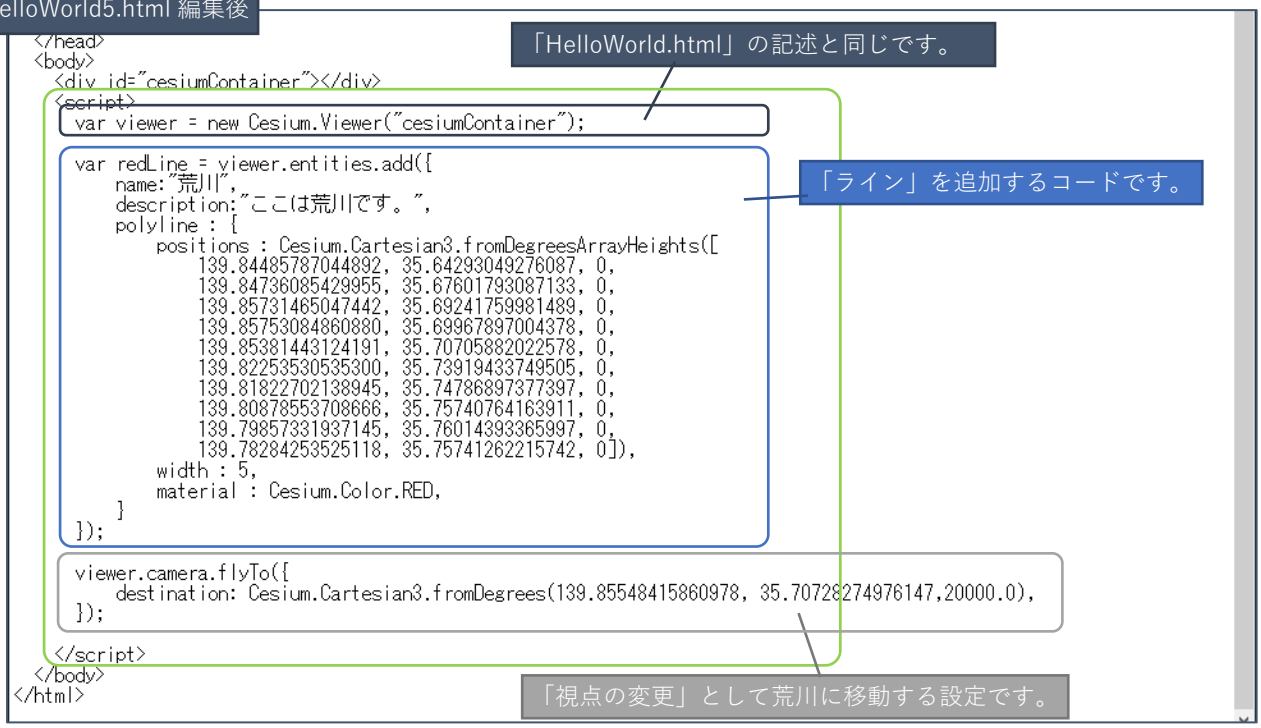
```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var redLine = viewer.entities.add({
          name:"荒川",
          description:"ここは荒川です。",
          polyline : {
              positions : Cesium.Cartesian3.fromDegreesArrayHeights([
                  139.84485787044892, 35.64293049276087, 0,
                  139.84736085429955, 35.67601793087133, 0,
                  139.85731465047442, 35.69241759981489, 0,
                  139.85753084860880, 35.69967897004378, 0,
                  139.85381443124191, 35.70705882022578, 0,
                  139.82253530535300, 35.73919433749505, 0,
                  139.81822702138945, 35.74786897377397, 0,
                  139.80878553708666, 35.75740764163911, 0,
                  139.79857331937145, 35.76014393365997, 0,
                  139.78284253525118, 35.75741262215742, 0]),
              width : 5,
              material : Cesium.Color.RED,
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.85548415860978, 35.70728274976147,20000.0),
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

この部分は２－３「視点の変更」で説明する内容ですが、
Cesium で表示する際に便利な部分であるため追加しています。

【編集部分の解説】

HelloWorld5.html 編集後




```
</head>
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer");

    var redLine = viewer.entities.add({
        name:"荒川",
        description:"ここは荒川です。",
        polyline : {
            positions : Cesium.Cartesian3.fromDegreesArrayHeights([
                139.84485787044892, 35.64293049276087, 0,
                139.84736085429955, 35.67601793087133, 0,
                139.85731465047442, 35.69241759981489, 0,
                139.85753084860880, 35.69967897004378, 0,
                139.85381443124191, 35.70705882022578, 0,
                139.82253530535300, 35.73919433749505, 0,
                139.81822702138945, 35.74786897377397, 0,
                139.80878553708666, 35.75740764163911, 0,
                139.79857331937145, 35.76014393365997, 0,
                139.78284253525118, 35.75741262215742, 0]),
            width : 5,
            material : Cesium.Color.RED,
        }
    });

    viewer.camera.flyTo({
        destination: Cesium.Cartesian3.fromDegrees(139.85548415860978, 35.70728274976147,20000.0),
    });

  </script>
</body>
</html>
```

【ラインを追加する部分のコード解説】

ラインを追加するためにはこれまでと同様に「veiwer.entities.add」メソッドを用います。

「veiwer.entities.add」で追加する際、表示する名称や説明文、ラインの形状などを設定します。

ラインの形状の設定は 4 行目の「polyline」内で行い、ラインを配置するポイント群の空間座標やラインの太さ、色などをメンバーの値に設定しています。各メンバーの解説については次頁の表の通りです。

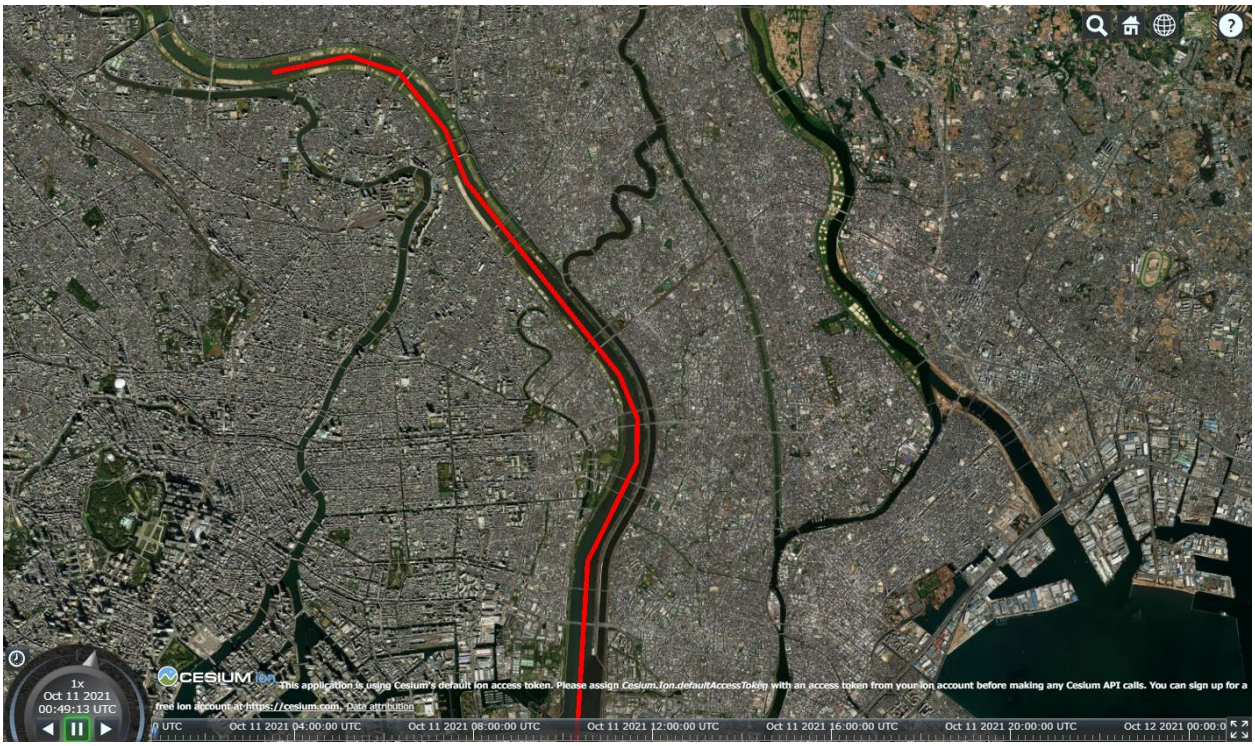
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.entities.add()・・・オブジェクトを追加するメソッドの呼び出し部。
2	name:	Cesium 上に表示されたラインをクリックした際に表示される情報窓のタイトルの設定部分。 例題では“荒川”と設定。
3	description:	Cesium 上に表示されたラインをクリックした際に表示される情報に記載する内容を設定。 例題では"ここは荒川です。"と設定。
4	polyline:	「ライン」を示すポイント群の配置や太さ、色などの設定部分。
5～15	positions:	「ライン」を配置するポイント群の空間座標の設定部分。 Cesium.Cartesian3.fromDegreesArrayHeights()メソッドを利用してポイントそれぞれの緯度、経度、ポイントを置く高さを指定している。 メソッドには緯度、経度、ポイントを置く高さの順にカンマ区切りで記述し、記述されたポイントの順番にラインがつながる。 例題では 10 ヵ所のポイントをつなげ、“荒川”の一部を示している。いずれも高さ＝0 で設定。
16	width:	配置する「ライン」の太さの設定部分。例題では 5 ピクセルで設定。
17	material:	配置する「ライン」の色の設定部分。例題では赤で設定。

⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld5.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld5.html」を表示させます。

表示させた結果は下図のようになります。



ラインをクリックすると右上に情報が表示されます。

【参考：複数のラインを追加したい場合】

複数のラインを追加したい場合、複数のポイントを追加したい場合と同様にラインを一つずつ追加します。

複数ラインの追加例

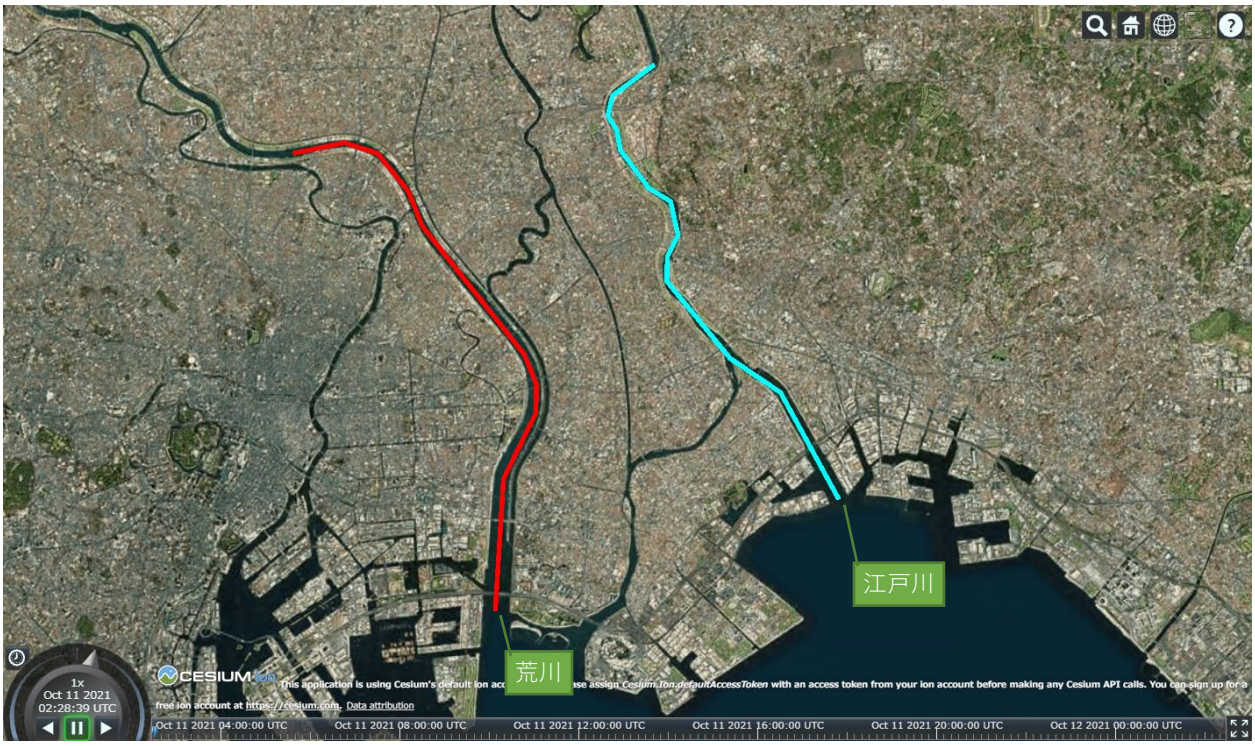
```
<div id="cesiumContainer"></div>
<script>
  var viewer = new Cesium.Viewer("cesiumContainer");

  var redLine = viewer.entities.add({
      name:"荒川",
      description:"ここは荒川です。",
      polyline : {
          positions : Cesium.Cartesian3.fromDegreesArrayHeights([
            139.8449, 35.6429, 0,
            139.8474, 35.6760, 0,
            139.8573, 35.6924, 0,
            139.8575, 35.6997, 0,
            139.8538, 35.7071, 0,
            139.8225, 35.7392, 0,
            139.8182, 35.7479, 0,
            139.8088, 35.7574, 0,
            139.7986, 35.7601, 0,
            139.7828, 35.7574, 0]),
          width : 5,
          material : Cesium.Color.RED
      }
  });

  var redLine = viewer.entities.add({
      name:"江戸川",
      description:"ここは江戸川です。",
      polyline : {
          positions : Cesium.Cartesian3.fromDegreesArrayHeights([
            139.9502, 35.6708, 0,
            139.9323, 35.6976, 0,
            139.9168, 35.7062, 0,
            139.8975, 35.7254, 0,
            139.8976, 35.7316, 0,
            139.9009, 35.7370, 0,
            139.8986, 35.7454, 0,
            139.8919, 35.7488, 0,
            139.8832, 35.7583, 0,
            139.8822, 35.7630, 0,
            139.8794, 35.7670, 0,
            139.8808, 35.7719, 0,
            139.8937, 35.7797, 0]),
          width : 5,
          material : Cesium.Color.fromBytes(0,255,255,255)
      }
  });

  viewer.camera.flyTo({
      destination: Cesium.Cartesian3.fromDegrees(139.8854,35.7025,30000.0),
  });
```

「荒川」のラインを追加するコード

「江戸川」にラインを追加するコード

色の指定方法を
Cesium.Color(red, green, blue, alpha)
の記述方法で設定したもの。記述の説明については『【補足資料３】色の設定部分の記述方法』を参照。

表示させた結果は次頁のようになります。



複数ラインを追加した html コードの記述例は「HelloWorld5_Example_EX.html」です。参考にしてください。

また、ラインとポイントをそれぞれ追加し、同一の画面上に表示させることも可能です。興味のある方は試してみてください。

以上で本項目は終了です。

２－３．初期視点の変更

[説明]

ここでは、Cesium で閲覧する際の初期視点を変更する方法について説明します。

Cesium では、視点の設定が指定されていない場合、北アメリカ大陸が表示されます。
各々で 3DTiles や KML データ、および「ポイント」や「ライン」、「ポリゴン」等を追加したデータを閲覧する際、表示させたい地域に自動的に視点が移動する処理を追加することで、用途に合った表示ができます。

実際に２－２[実践１]でポイントを追加した「HelloWorld3.html」および２－２[実践３]でポリゴンを追加した「HelloWorld6.html」を利用して、視点を移動させる方法を実践します。

[実践１]　真上からの視点

ここでは２－２[参考１]でポイントを追加した「HelloWorld4.html」を利用し、真上から見た視点の変更方法について試します。
この章で挙げる html コードの記述例は「HelloWorld6_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld4.html」をコピー＆ペーストし、ファイル名を「HelloWorld6.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld6.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld6.html 編集前

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var point = viewer.entities.add({
          name:"国立競技場",
          description:"〒160-0013 東京都新宿区霞ヶ丘町１０−１",
          position : Cesium.Cartesian3.fromDegrees(139.7145,35.6779,0),
          point : {
              pixelSize : 10,
              color : Cesium.Color.BLUE
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.7145,35.6779,3000.0)
      });

    </script>
  </body>
</html>
```

この赤線部分を下図のように編集します

編集後

```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var point = viewer.entities.add({
          name:"国立競技場",
          description:"〒160-0013 東京都新宿区霞ヶ丘町１０−１",
          position : Cesium.Cartesian3.fromDegrees(139.7145,35.6779,0),
          point : {
              pixelSize : 10,
              color : Cesium.Color.BLUE
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.7113,35.6812,1500.0)
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

【視点変更部分の解説】

HelloWorld6.html 変種後

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var point = viewer.entities.add({
          name:"国立競技場",
          description:"〒160-0013 東京都新宿区霞ヶ丘町１０−１",
          position : Cesium.Cartesian3.fromDegrees(139.7145,35.6779,0),
          point : {
              pixelSize : 10,
              color : Cesium.Color.BLUE
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.7113,35.6812,1500.0)
      });

    </script>
  </body>
</html>
```

「ポイント」を追加した部分。

視点が千駄ヶ谷駅に移動するように緯度と経度を変更しています。
また高度を低い位置に変更しています。

【視点変更部分のコード解説】

視点を変更するためには「viewer.camera.flyTo」メソッドを用います。またメンバーの「description」で視点の緯度、経度、視点の高さを設定します。「description」の解説については下表の表の通りです。

（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.camera.flyTo()・・・視点を操作するメソッドの呼び出し部。
2	description:	視点の基準位置の設定部分。Cesium.Cartesian3.fromDegrees()メソッドを利用して、緯度、経度、視点の高さを指定している。 メソッドには緯度、経度、視点の高さの順にカンマ区切りで記述する。 例題では千駄ヶ谷駅の緯度、経度が設定されている。

⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld6.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld6.html」を表示させます。

表示させた結果は下図のようになります。

画面の中央に「千駄ヶ谷駅」が表示され、また低い高度から見た視点に変更されています。

興味のある方は緯度、経度、視点の高さを好きな場所や高度に変更し、試してみてください。

以上で本項目は終了です。

[実践２] 斜め上からの視点

Cesium の特性上、真上から見下ろした視点のほか、水平方向の視点や斜め上からの視点など様々な角度からの視点を設定することができます。

ここでは２－２[実践３]でポリゴンを追加した「HelloWorld3.html」を利用し、斜め上からの視点に変更する方法について試します。

この章で挙げる html コードの記述例は「HelloWorld7_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld3.html」をコピー＆ペーストし、ファイル名を「HelloWorld7.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld7.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld7.html 編集前

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var polygon = viewer.entities.add({
          name:"東京都庁",
          description:"ここは東京都庁です。",
          polygon : {
              hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                  139.69163740881, 35.68903147599, 0,
                  139.69143377086, 35.68993508559, 0,
                  139.69184361318, 35.68999687609, 0,
                  139.69203616751, 35.68909180654, 0]),
              extrudedHeight: 200.0,
              material : Cesium.Color.RED.withAlpha(0.5),
              outline : true,
              outlineColor : Cesium.Color.BLACK
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.691734969185, 35.689513446065, 1000.0),
      });

    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後

```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var polygon = viewer.entities.add({
          name:"東京都庁",
          description:"ここは東京都庁です。",
          polygon : {
              hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                  139.69163740881, 35.68903147599, 0,
                  139.69143377086, 35.68993508559, 0,
                  139.69184361318, 35.68999687609, 0,
                  139.69203616751, 35.68909180654, 0]),
              extrudedHeight: 200.0,
              material : Cesium.Color.RED.withAlpha(0.5),
              outline : true,
              outlineColor : Cesium.Color.BLACK
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.6995, 35.6836, 1000.0),
          orientation : {
              heading : Cesium.Math.toRadians(-45.0),
              pitch : Cesium.Math.toRadians(-45.0),
              roll : 0.0
          }
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

【視点変更部分の解説】

HelloWorld7.html 編集後

```
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer");

    var polygon = viewer.entities.add({
        name:"東京都庁",
        description:"ここは東京都庁です。",
        polygon : {
            hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                139.69163740881, 35.68903147599, 0,
                139.69143377086, 35.68993508559, 0,
                139.69184361318, 35.68999687609, 0,
                139.69203616751, 35.68909180654, 0]),
            extrudedHeight: 200.0,
            material : Cesium.Color.RED.withAlpha(0.5),
            outline : true,
            outlineColor : Cesium.Color.BLACK
        }
    });

    viewer.camera.flyTo({
        destination: Cesium.Cartesian3.fromDegrees(139.6995, 35.6836, 1000.0),
        orientation : {
            heading : Cesium.Math.toRadians(-45.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
        }
    });

  </script>
</body>
</html>
```

「ポリゴン」を追加した部分。

視点の向きに伴い、基準位置の緯度経度を変更する。

北向き（0 度）から-45 度（北西）を向く設定です。

水平（0 度）から-45 度（下方向）を向く設定です。

緯度経度や角度などの空間座標の求め方については、巻末の
『【補足資料１】視点の緯度や経度、角度などの求め方』を参照。

【視点変更部分のコード解説】

真上からの視点変更と同様に「viewer.camera.flyTo」メソッドを用いますが、「orientation」のメンバーを追加し視点の方向（縦・横方向およびロール角）を設定します。「description」および「orientation」の解説については下表の表の通りです。

（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.camera.flyTo()・・・視点を操作するメソッドの呼び出し部。
2	description:	視点の基準位置の設定部分。Cesium.Cartesian3.fromDegrees()メソッドを利用して、緯度、経度、視点の高さを指定している。 メソッドには緯度、経度、視点の高さの順にカンマ区切りで記述する。
3	orientation:	視点の方向の設定部分。
4	heading:	視点の横方向（東西南北の方向）の設定部分。Cesium.Math.toRadians()メソッドを利用して方向を指定している。0 の場合は北、90（または-270）の場合は東を指し示す。
5	pitch:	視点の縦方向（天地の向き）の設定部分。Cesium.Math.toRadians()メソッドを利用して方向を指定している。0 の場合は水平線と平行の向きとなり、90（または-270）の場合は真上、-90（または 270）の場合は真下を指し示す。
6	roll:	視点のロール角の設定部分。0 の場合は水平に並行な視点となる。0 以外の値を用いるシーンはほぼ無いため、詳細は割愛する。

⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld7.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld7.html」を表示させます。

表示させた結果は下図のようになります。

真上から見下ろす視点であった「HelloWorld3.html」と比べ、ポリゴンの立体感をすぐに確認することができます。

【斜め上からの視点に変更する際の注意点】

視点の変更は２行目の「description」で設定した空間座標を基準とし、「orientation」の「heading」で横方向（東西南北の方位）を、「pitch」で縦方向（天地の方向）設定しています。

「heading」や「pitch」は角度を設定しているだけですので、視点となる基準点の空間座標（緯度、経度、視点の高さ）も変更する必要があります。

用途に合わせて、視点の変更と空間座標の調整を行ってください。

空間座標の求め方については、巻末の『【補足資料１】視点の緯度や経度、角度などの求め方』に記載しています。参考にしてください。

3．技術テクニック：後編

ここでは、『GIS 実習オープン教材：Cesium 入門』で紹介されていないプラスアルファの技術テクニック（下表４～７）を説明します。

	章	技術テクニック	説明
1	2-1	機能のオン／オフ	Cesiumに標準搭載されている基本機能のオン／オフの切り替え方法について説明します。
2	2-2	レイヤの追加	空間座標を利用し、ポイント、ライン、ポリゴンの各追加方法について説明します。
3	2-3	視点の変更	Cesiumで閲覧する際の初期視点の設定方法について説明します。
4	3-1	ラベルの追加	空間座標を利用し、ラベルの追加方法について説明します。
5	3-2	ベースマップの 地形の変更	Cesiumで閲覧する際に読み込むベースマップの地形の変更方法について説明します。
6	3-3	ホームボタンの 設定変更	Cesiumに標準搭載されているホームボタンで設定される空間情報の変更方法について説明します。
7	3-4	物件の情報による 色の変更	物件が所有している情報にあわせた色の変更方法について説明します。

具体的な手法の説明は、前編２－３[実習２]で作成した「HelloWorld7.html」および、研修用のサンプルデータである「chino_3dtiles.html」を利用しながら説明していきます。

３－１．ラベルの追加

[説明]

ここでは、Cesium 上に「ラベル」を追加する方法について説明します。

「ラベル」も空間座標データとして扱うことで、「ポイント」や「ポリゴン」と同様に追加することができます。

[実践]

例題として、２－３[実践２]の東京都庁のポリゴン上にラベルを追加することを実践します。

なお、この章で挙げる html コードの記述例は「HelloWorld8_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld7.html」をコピー＆ペーストし、ファイル名を「HelloWorld8.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld8.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld8.html 編集前

```
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer");

    var polygon = viewer.entities.add({
        name:"東京都庁",
        description:"ここは東京都庁です。",
        polygon : {
            hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                139.69163740881, 35.68903147599, 0,
                139.69143377086, 35.68993508559, 0,
                139.69184361318, 35.68999687609, 0,
                139.69203616751, 35.68909180654, 0]),
            extrudedHeight: 200.0,
            material : Cesium.Color.RED.withAlpha(0.5),
            outline : true,
            outlineColor : Cesium.Color.BLACK
        }
    });

    viewer.camera.flyTo({
        destination: Cesium.Cartesian3.fromDegrees(139.6995, 35.6836, 1000.0),
        orientation : {
            heading : Cesium.Math.toRadians(-45.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
        }
    });

  </script>
</body>
</html>
```

この部分に下図のコードを追加します。

編集後

```
<head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");

      var polygon = viewer.entities.add({
          name:"東京都庁",
          description:"ここは東京都庁です。",
          polygon : {
              hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                  139.69163740881, 35.68903147599, 0,
                  139.69143377086, 35.68993508559, 0,
                  139.69184361318, 35.68999687609, 0,
                  139.69203616751, 35.68909180654, 0]),
              extrudedHeight: 200.0,
              material : Cesium.Color.RED.withAlpha(0.5),
              outline : true,
              outlineColor : Cesium.Color.BLACK,
          },
      });

      var label = viewer.entities.add({
          position: Cesium.Cartesian3.fromDegrees(139.6917, 35.6895, 250),
          label: {
              text: "東京都庁",
              font: "20px sans-serif",
              fillColor: Cesium.Color.YELLOW,
              outlineColor: Cesium.Color.BLACK,
              outlineWidth: 2,
              style: Cesium.LabelStyle.FILL_AND_OUTLINE
          }
      });

      viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(139.6995, 35.6836, 1000.0),
          orientation : {
              heading : Cesium.Math.toRadians(-45.0),
              pitch : Cesium.Math.toRadians(-45.0),
              roll : 0.0
          }
      });

    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

【編集部分の解説】

HelloWorld8.html 編集後

```
                143377086, 35.68993508559, 0,
                139.69184361318, 35.68999687609, 0,
                139.69203616751, 35.68909180654, 0]),
            extrudedHeight: 200.0,
            material : Cesium.Color.RED.withAlpha(0.5),
            outline : true,
            outlineColor : Cesium.Color.BLACK,
        },
});

var label = viewer.entities.add({
    position: Cesium.Cartesian3.fromDegrees(139.6917, 35.6895, 250),
    label: {
        text: "東京都庁",
        font: "20px sans-serif",
        fillColor: Cesium.Color.YELLOW,
        outlineColor: Cesium.Color.BLACK,
        outlineWidth: 2,
        style: Cesium.LabelStyle.FILL_AND_OUTLINE
    }
});

viewer.camera.flyTo({
    destination: Cesium.Cartesian3.fromDegrees(139.6995, 35.6836, 1000.0),
    orientation : {
        heading : Cesium.Math.toRadians(-45.0),
```

ラベルの位置（空間座標）とラベルの設定を行っている。

【ラベルを追加する部分のコード解説】

ラベルを追加するためには「veiwer.entities.add」メソッドを用います。また「veiwer.entities.add」で追加する際、ポイントの空間座標やポイントの大きさ、色などをメンバーの値に設定することで、用途に合ったポイントとして追加することができます。各メンバーの解説については下表の通りです。

（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.entities.add()・・・オブジェクトを追加するメソッドの呼び出し部。
2	position:	「ラベル」を配置する空間座標の設定部分。Cesium.Cartesian3.fromDegrees()メソッドを利用して、緯度、経度、ラベルを置く高さを指定している。 メソッドには緯度、経度、ラベルを置く高さの順にカンマ区切りで記述する。
3	label:	表示する「ラベル」の設定部分。
4	text:	表示する文字の設定部分。例題では“東京都庁”で設定。
5	font:	表示するラベルのフォントの設定部分。フォントサイズとフォントを設定する。省略可。省略されている場合、デフォルト値＝”30px sans-serif”で処理される。
6	fillColor:	表示するラベルの色の設定部分。省略可。省略されている場合、デフォルト値＝”WHITE”で処理される。
7	outlineColor:	表示するラベルの枠の色の設定部分。省略可。省略されている場合、デフォルト値＝”BLACK”で処理される。但し、9 行目の「style:」の設定が「FILL」の場合は無効となる。
8	outlineWidth:	表示するラベルの枠の太さの設定部分。省略可。省略されている場合、デフォルト値＝”1.0”で処理される。但し、9 行目の「style:」の設定が「FILL」の場合は無効となる。
9	style:	表示するラベルのスタイルの設定部分。FILL（枠無し文字）、OUTLINE（枠だけの文字）、FILL_AND_OUTLINE（枠あり文字）で設定する。 省略可。省略されている場合、デフォルト値＝”FILL”で処理される。

⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld8.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld8.html」を表示させます。

表示させた結果は下図のようになります。

興味のある方はラベルの色や配置する視点の高さなどを変更し、試してみてください。

以上で本項目は終了です。

３－２．ベースマップの地形の変更

[説明]

ここでは、Cesium で利用するベースマップの地形の変更方法について説明します。
また、補足として、ベースマップに表示される地図の種類の変更についても説明します。

Cesium を利用する際、デフォルトの設定のままでは、「WGS84 Ellipsoid」が使用され、平面地形で表示されます。
しかし､本研修で試用する 3D Tiles データや KML データなど標高の情報が含まれるデータを扱う場合、3D の地形で表示される「Cesium World Terrain」に変更した方が視認性や利便性が高くなります。
このように、用途に応じて地形を変更する必要が出てきますが、Cesium を利用する際に予め 3D の地形で表示されるように指定することができます。

[実践]

例題として、研修用サンプルデータの中から長野県茅野市の建物の 3D Tiels データを表示する「chino_3dtiles.html」を利用します。

この「chino_3dtiles.html」を Cesium で表示した場合、ベースマップの地形が「WGS84 Ellipsoid」であるため、建物（3D Tiels データ）が宙に浮いた状態で表示されます。この地形を「Cesium World Terrain」に変更することで建物は地形にあった位置に表示されるようになります。(詳しくは実習テキスト（基礎編）「２－８．ブラウザ上で Cesium を利用したデータを確認する」を参照してください。)

この章で挙げる html コードの記述例は「chino_3dtiles2_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「chino_3dtiles.html」をコピー＆ペーストし、ファイル名を「chino_3dtiles2.html」に変更します。
（作業イメージは割愛）

---

③ 「chino_3dtiles2.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

chino_3dtiles2.html 編集前

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");
      var tileset1 = new Ces      Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.      (function(tileset1) {
        viewer.scene.primiti      .add(tileset1);
        tileset1.style = new      sium.Cesium3DTileStyle({
          color: 'rgb(0, 0,      )',
          markerSymbol: '?'
        });
        viewer.camera.flyTo(
          destination: Cesiu      artesian3.fromDegrees(138.2, 35.90, 10000),
          orientation : {
            heading : Cesium      h.toRadians(0.0),
            pitch : Cesium.M      toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(      or) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後

```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer", {
          terrainProvider: Cesium.createWorldTerrain(),
          imageryProvider: new Cesium.OpenStreetMapImageryProvider({
              url : 'https://a.tile.openstreetmap.org/'
          })
      });

      var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.then(function(tileset1) {
        viewer.scene.primitives.add(tileset1);
        tileset1.style = new Cesium.Cesium3DTileStyle({
          color: 'rgb(0, 0, 255)',
          markerSymbol: '?'
        });
        viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(138.2, 35.90, 10000),
          orientation : {
            heading : Cesium.Math.toRadians(0.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(error) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

ソースコードの解説は次頁を参照

【編集部分の解説】

chino_3dtiles2.html 編集後

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer", {
          terrainProvider: Cesium.createWorldTerrain(),
          imageryProvider: new Cesium.OpenStreetMapImageryProvider({
              url : 'https://a.tile.openstreetmap.org/'
          })
      });

      var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.then(function(tileset1) {
        viewer.scene.primitives.add(tileset1);
        tileset1.style = new Cesium.Cesium3DTileStyle({
          color: 'rgb(0, 0, 255)',
          markerSymbol: '?'
        });
        viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(138.2, 35.90, 10000),
          orientation : {
            heading : Cesium.Math.toRadians(0.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(error) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

地形の設定と地図の種類の設定を行っている。

長野県茅野市の 3D Tiles データを表示する部分。

【地形および地図の種類を変更する部分のコード解説】

地形および地図の種類を変更するには「Cesium.Viewer」メソッド内でメンバーの「terrainProvider」および「imageryProvider」の設定を行います。各メンバーの解説については下表の通りです。

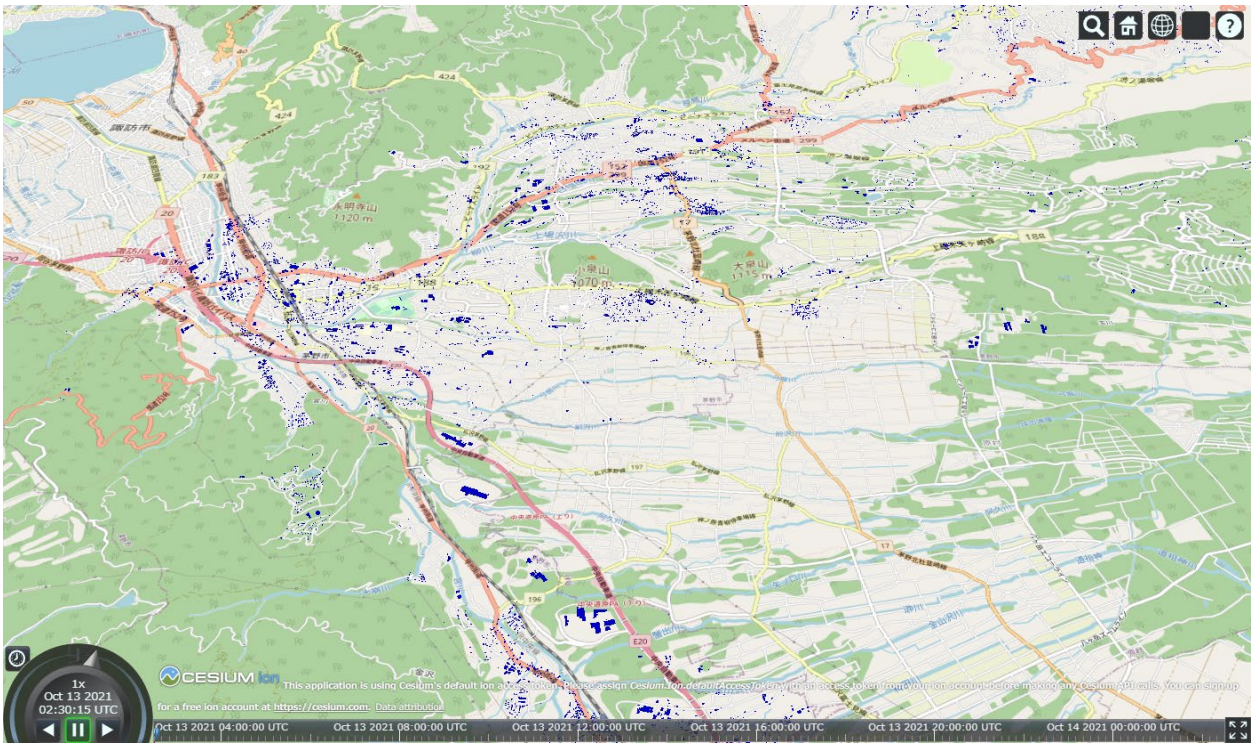
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	Cesium.Viewer()・・・ビューアを読み込むメソッドの呼び出し部。
2	terrainProvider:	読み込む「地形（terrain）」の設定部分。「Cesium.createWorldTerrain()」と指定することでベースマップの地形が 3D 地形となる。 省略可。省略している場合は平面地形となる。
2	imageryProvider:	読み込む「地図の種類」の設定部分。省略可。 省略している場合は「EllipsoidTerrainProvider()」が読み込まれる。 例題では OpenStreetMap の地図を読み込む設定。 設定できる地図の種類については下記の Cesium 公式のドキュメントを参照のこと。 https://cesium.com/learn/cesiumjs/ref-doc/ImageryProvider.html
3	url:	読み込む地図の URL。 上記の imageryProvider で設定した地図の種類に従って設定する。

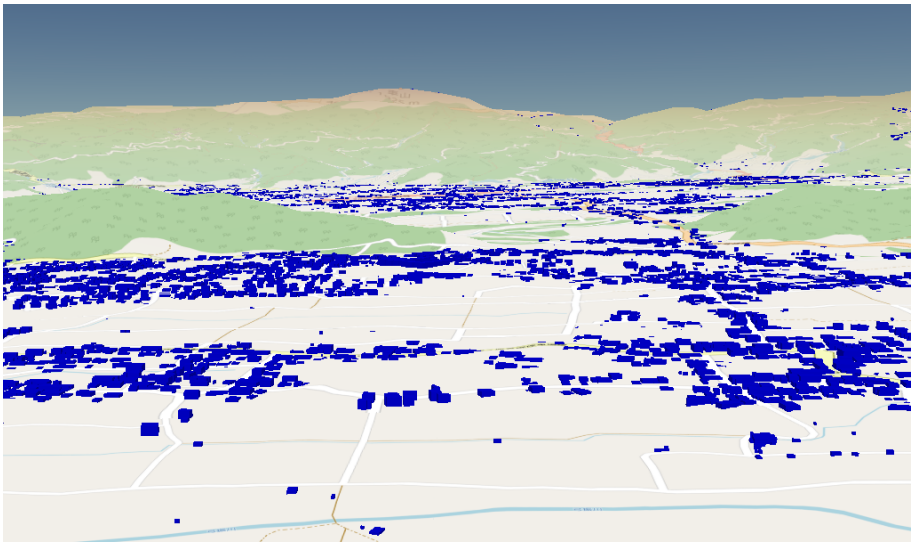
⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「chino_3dtiles2.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「chino_3dtiles2.html」を表示させます。

表示させた結果は下図のようになります。



マウスで視点を動かすと、3D 地形で表示されていることがわかります。



以上で本項目は終了です。

３－３．ホームボタンの設定変更

[説明]

ここでは、Cesium の基本機能で表示される「ホームボタン」の変更方法について説明します。

「ホームボタン」をクリックすると、指定した空間座標に移動できますが、デフォルトの設定のままでは、北アメリカ大陸に移動します。

「ホームボタン」で移動する空間座標は、(2021.10.13 現在）変更することができないため、「ホームボタン」をクリックしたときのイベントを制御し、指定する空間座標に移動するように処理を追加します。

そうすることにより、疑似的に「ホームボタン」をクリックしたときの移動先を自由に設定させることができます。

[実践]

例題として、「HelloWorld8.html」を利用し、「ホームボタン」をクリック後、東京都庁から富士山に移動する設定を実践します。

この章で挙げる html コードの記述例は「HelloWorld9_Example.html」を参考にしてください。

なお、前章の「HelloWorld8.html」を Cesium で表示し、「ホームボタン」をクリックすると、北アメリカ大陸に移動しますので、興味のある方は試してみてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「HelloWorld8.html」をコピー＆ペーストし、ファイル名を「HelloWorld9.html」に変更します。
（作業イメージは割愛）

---

③ 「HelloWorld9.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

HelloWorld9.html 編集前

```
        padding: 0;
        overflow: hidden;
    }
  </style>
</head>
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer");

    var polygon = viewer.e    ties.add({
        name:"東京都庁",
        description:"ここ(    京都庁です。",
        polygon : {
            hierarchy : Ce    m.Cartesian3.fromDegreesArrayHeights([
                139.691637    81, 35.68903147599, 0,
                139.691433    86, 35.68993508559, 0,
                139.691843    18, 35.68999687609, 0,
                139.692036    51, 35.68909180654, 0]),
            extrudedHeight    00.0,
            material : Ces    .Color.RED.withAlpha(0.5),
            outline : true
            outlineColor :    sium.Color.BLACK,
        },
    });
```

この部分を下図のように編集します

編集後

```
    margin: 0;
    padding: 0;
    overflow: hidden;
  }
  </style>
</head>
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer", {
        terrainProvider: Cesium.createWorldTerrain()
    });

    viewer.homeButton.viewModel.command.beforeExecute.addEventListener(
        function(e) {
            e.cancel = true;
            viewer.camera.flyTo({
                destination: Cesium.Cartesian3.fromDegrees(138.7308, 35.2629, 5000.0),
                orientation : {
                    heading : Cesium.Math.toRadians(0.0),
                    pitch : Cesium.Math.toRadians(-15.0),
                    roll : 0.0
                }
            });
        }
    );

    viewer.entities.add({
       position: Cesium.Cartesian3.fromDegrees(138.7311, 35.3629, 4000),
       label: {
         text: "富士山"
       }
    });

    var polygon = viewer.entities.add({
        name:"東京都庁",
        description:"ここは東京都庁です。",
        polygon : {
            hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                139.69163740881, 35.68903147599, 0,
                139.69143377086, 35.68993508559, 0,
                139.69184361318, 35.68999687609, 0,
                139.69203616751, 35.68909180654, 0]),
            extrudedHeight: 200.0,
            material : Cesium.Color.RED.withAlpha(0.5),
            outline : true,
            outlineColor : Cesium.Color.BLACK,
        },
    });
```

ソースコードの解説は次頁を参照

【編集部分の解説】

HelloWorld9.html 編集後

```
        padding: 0;
        overflow: hidden;
    }
  </style>
</head>
<body>
  <div id="cesiumContainer"></div>
  <script>
    var viewer = new Cesium.Viewer("cesiumContainer", {
        terrainProvider: Cesium.createWorldTerrain()
    });

    viewer.homeButton.viewModel.command.beforeExecute.addEventListener(
        function(e) {
            e.cancel = true;
            viewer.camera.flyTo({
                destination: Cesium.Cartesian3.fromDegrees(138.7308, 35.2629, 5000.0),
                orientation : {
                    heading : Cesium.Math.toRadians(0.0),
                    pitch : Cesium.Math.toRadians(-15.0),
                    roll : 0.0
                }
            });
        }
    );

    viewer.entities.add({
       position: Cesium.Cartesian3.fromDegrees(138.7311, 35.3629, 4000),
       label: {
         text: "富士山"
       }
    });

    var polygon = viewer.entities.add({
        name:"東京都庁",
        description:"ここは東京都庁です。",
        polygon : {
            hierarchy : Cesium.Cartesian3.fromDegreesArrayHeights([
                139.69163740881, 35.68903147599, 0,
                139.69143377086, 35.68993508559, 0,
                139.69184361318, 35.68999687609, 0,
                139.69203616751, 35.68909180654, 0]),
            extrudedHeight: 200.0,
            material : Cesium.Color.RED.withAlpha(0.5),
            outline : true,
            outlineColor : Cesium.Color.BLACK,
        },
    });
```

ベースマップの地形を 3D 地形に変更するコードを追加しています。（本誌３－２参照）

ホームボタンの設定変更を追加するコードです。

「富士山」のラベルを追加するコードを追加しています。（本誌３－１参照）

東京都庁のポリゴンを追加したコードです。（本資料２－２[実践３]参照）

【ホームボタンの設定変更を追加する部分のコード解説】

先に述べた通り、“ホームボタンをクリックした”イベントが発生した時、指定する空間座標に移動するように処理を追加しています。

青枠内１行目の「viewer.homeButton.viewModel.command.beforeExecute.addEventListener」でホームボタンをクリックしたときのイベントを検知し、２行目以降で移動する処理を実行しています。

各メンバーの解説については下表の通りです。

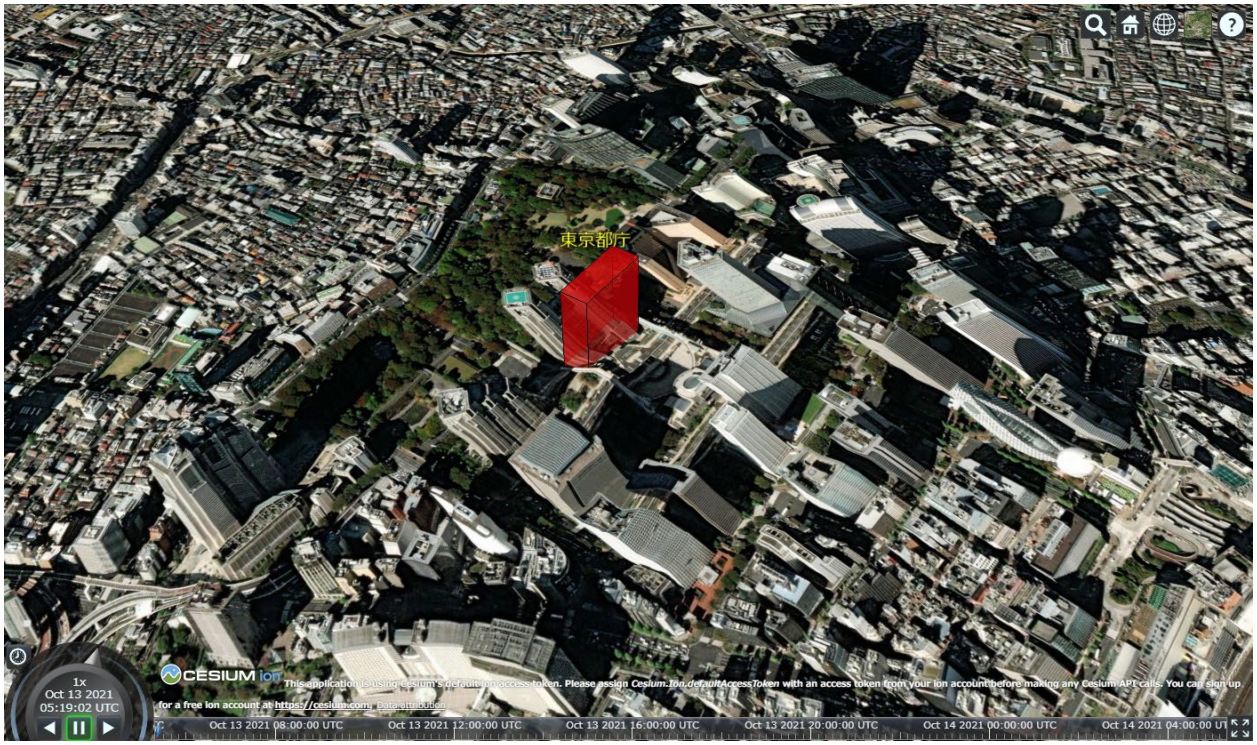
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	viewer.homeButton.viewModel.command.beforeExecute.addEventListen() ・・・ホームボタンをクリックした際のイベント（メソッド）の呼び出し部。
2	（オブジェクト）	function(e){} ・・・viewer.homeButton.viewModel.command.beforeExecute.addEventListen()のイベントオブジェクト。イベント発生時に実行する処理の宣言部。
3	（オブジェクト）	e.cancel = true; ・・・イベントオブジェクトの有効／無効を設定する。true=有効で設定する。 なお、false=無効に設定した場合、本イベント自体が無効となり、ホームボタンをクリックした際は北アメリカ大陸を指す。
4	（メソッド）	viewer.camera.flyTo()・・・視点を操作するメソッドの呼び出し部。
5	description:	視点の基準位置の設定部分。Cesium.Cartesian3.fromDegrees()メソッドを利用して、緯度、経度、視点の高さを指定している。 メソッドには緯度、経度、視点の高さの順にカンマ区切りで記述する。
6	orientation:	視点の方向の設定部分。
7	heading:	視点の横方向（東西南北の方向）の設定部分。Cesium.Math.toRadians()メソッドを利用して方向を指定している。0 の場合は北向き、90（または-270）の場合は東を指し示す。
8	pitch:	視点の縦方向（天地の向き）の設定部分。Cesium.Math.toRadians()メソッドを利用して方向を指定している。0 の場合は水平線と平行の向きとなり、90（または-270）の場合は真上、-90（または 270）の場合は真下を指し示す。
9	roll:	視点のロール角の設定部分。0 の場合は水平に並行な視点となる。0 以外の値を用いるシーンはほぼ無いため、詳細は割愛する。

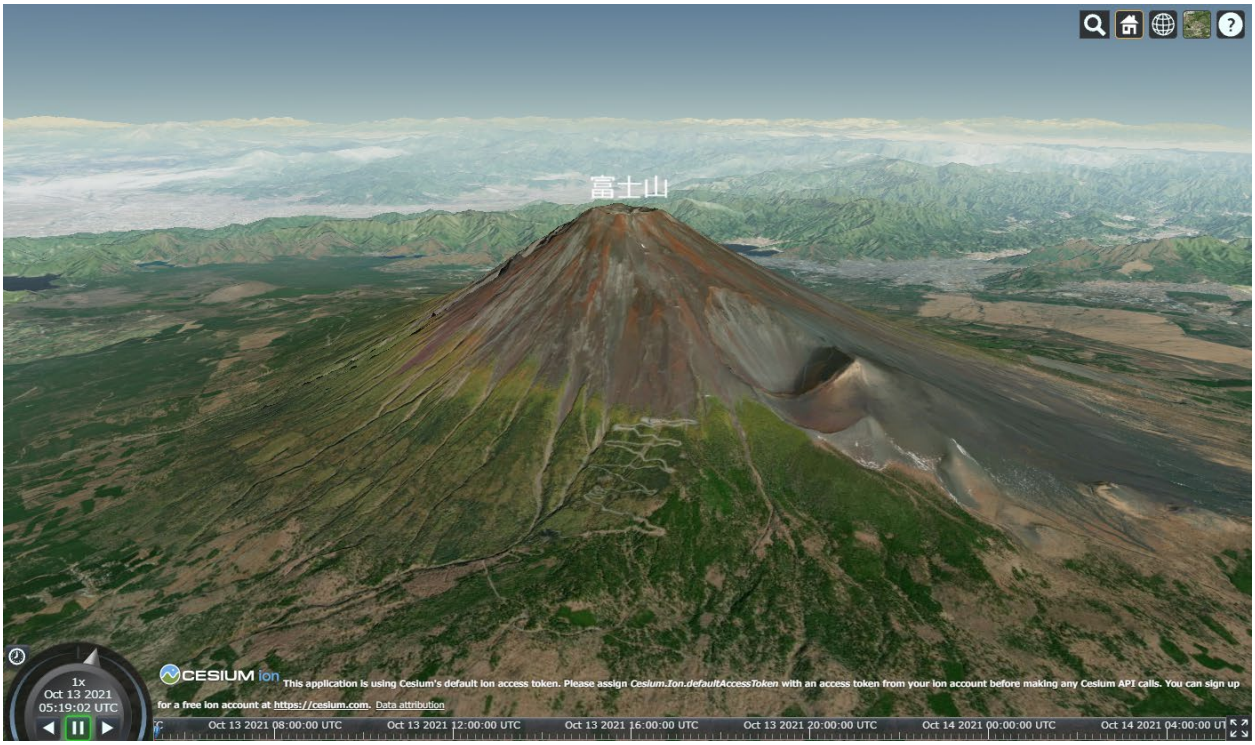
⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「HelloWorld9.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「HelloWorld9.html」を表示させます。

表示させた結果は下図のようになります。



ホームボタンをクリックすると富士山が表示されます。



以上で本項目は終了です。

３－４．物件の情報による色の変更

[説明]

3D Tiels データには様々な情報が付与されています。その情報を分類し、分類毎に色分けして表示することができれば視認性や利便性が高まります。

そこで、3D Tiels データを Cesium で利用する際、3D Tiels データに登録されている情報により物件を分類し、分類毎に色分けし表示する方法について説明します。

[実践]

例題として、研修用サンプルデータの中から長野県茅野市の建物の 3D Tiels データを表示する「chino_3dtiles.html」を利用します。
付与されている建物の階数で分類し、色分けを行います。

この章で挙げる html コードの記述例は「chino_3dtiles3_Example.html」を参考にしてください。

---

① 本書２－１①と同様に、リポジトリ（kensyu）内の【Apps】フォルダを開きます。
（作業イメージは割愛）

---

② 「chino_3dtiles.html」をコピー＆ペーストし、ファイル名を「chino_3dtiles3.html」に変更します。
（作業イメージは割愛）

---

③ 「chino_3dtiles3.html」をメモ帳で開きます。

④ 下図に従って、内容を編集し、上書き保存します。

chino_3dtiles3.html 編集前

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer");
      var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.then(function(tileset1) {
        viewer.scene.primitives.add(tileset1);
        tileset1.style = new Cesium.Cesium3DTileStyle({
          color: 'rgb(0, 0, 255)',
          markerSymbol: '?'
        });
        viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(138.2, 35.90, 10000),
          orientation : {
            heading : Cesium.Math.toRadians(0.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(error) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

この部分を下図のように編集します

編集後

```
        margin: 0;
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer", {
          terrainProvider: Cesium.createWorldTerrain()
      });

      var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.then(function(tileset1) {
        viewer.scene.primitives.add(tileset1);
        tileset1.style = new Cesium.Cesium3DTileStyle({
          color: {
            conditions:[
              ["Number(${地上階数}) >= 4","rgba(255, 0, 0, 0.5)"],
              ["Number(${地上階数}) >= 3","rgba(0, 255, 0,0.5)"],
              ["Number(${地上階数}) >= 2","rgba(0, 0, 255,0.5)"],
              ["true","rgb(255, 255, 255)"]
            ]
          },
          markerSymbol: '?'
        });

        viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(138.2, 35.90, 10000),
          orientation : {
            heading : Cesium.Math.toRadians(0.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(error) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

併せて、ベースマップの地形の設定も変更します

ソースコードの解説は次頁を参照

【編集部分の解説】

chino_3dtiles3.html 編集後

```
        padding: 0;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="cesiumContainer"></div>
    <script>
      var viewer = new Cesium.Viewer("cesiumContainer", {
          terrainProvider: Cesium.createWorldTerrain()
      });

      var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
      tileset1.readyPromise.then(function(tileset1) {
        viewer.scene.primitives.add(tileset1);
        tileset1.style = new Cesium.Cesium3DTileStyle({
          color: {
            conditions:[
              ["Number(${地上階数}) >= 4","rgba(255, 0, 0, 0.5)"],
              ["Number(${地上階数}) >= 3","rgba(0, 255, 0,0.5)"],
              ["Number(${地上階数}) >= 2","rgba(0, 0, 255,0.5)"],
              ["true","rgb(255, 255, 255)"]
            ]
          },
          markerSymbol: '?'
        });

        viewer.camera.flyTo({
          destination: Cesium.Cartesian3.fromDegrees(138.2, 35
          orientation : {
            heading : Cesium.Math.toRadians(0.0),
            pitch : Cesium.Math.toRadians(-45.0),
            roll : 0.0
          }
        });
      }).otherwise(function(error) {
        console.log(error);
      });
    </script>
  </body>
</html>
```

ベースマップの地形を 3D 地形に変更する
コードを追加しています。（本誌３－２参照）

色分けは上の行から順に分類される。従って、
4 階以上は赤
3 階以上は緑
2 階以上は青
上記以外（2 階未満）は白
と設定されていることになる。

【物件の情報の条件に従って配色を変更する部分のコード解説】

青枠 1 行目の「Cesium.Cesium3DTileStyle」メソッドを用いて 3DTiles データのスタイルを設定することができます。同 3 行目の「conditions:」で分類する条件と色の設定を行います。

「conditions:」に登録する条件と色の組合せは配列型で記述します。読み込んだ 3DTiles データを分類する処理は、記述した順番で処理され、合致した条件の配色が適用されます。

（例）

読み込んだデータが 3 階の情報をもっていた場合、始めに 1 つ目の“4 階以上は赤”に合致するか否かを処理します。

その結果は合致しないため、2 つ目の条件である“3 階以上は緑”に合致するか否かを処理します。

この結果は合致するため、“3 階以上は緑”の配色が適用されます。

どの条件にも合致しなかった場合、7 行目の「["true","rgb(255, 255, 255)"]」が適用されます。

各メンバーの解説については下表の通りです。

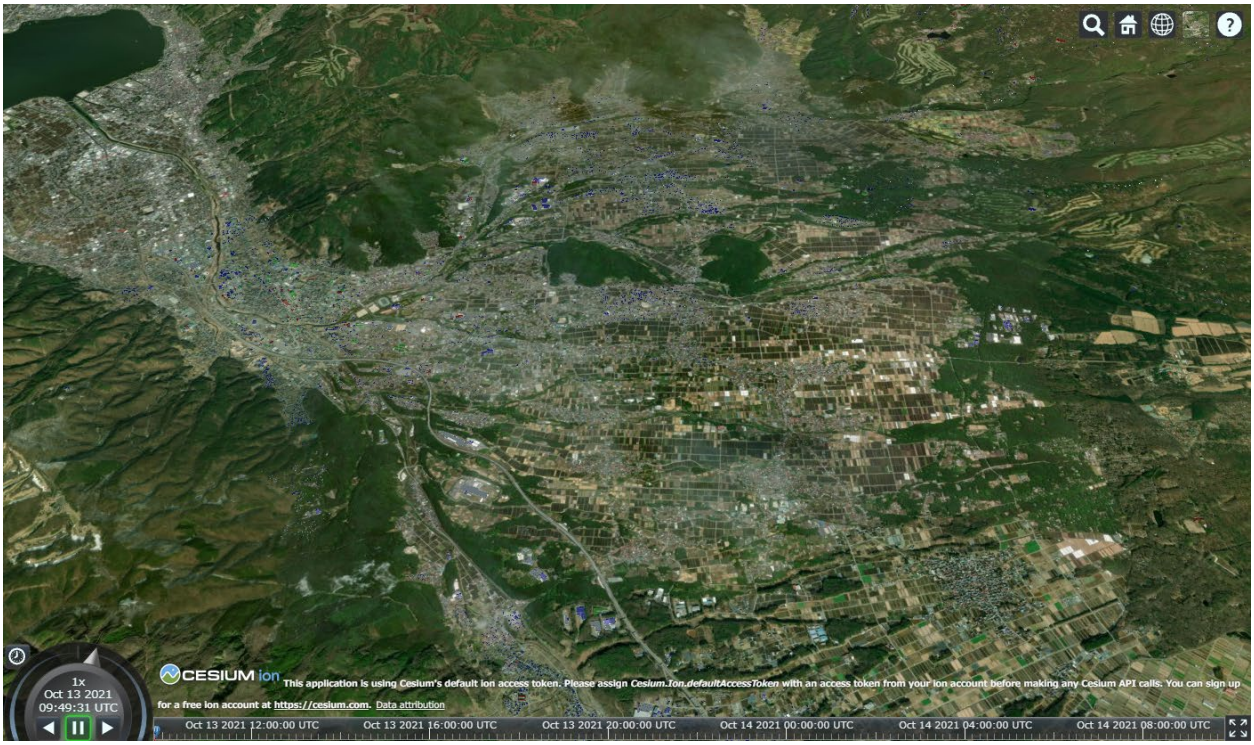
（行番号は上記コードの青枠内の行番号です）

行番号	メンバー	解説
1	（メソッド）	Cesium.Cesium3DTileStyle() ・・・3DTiles データのスタイルを設定するメソッドの呼び出し部。
2	color:	3DTiles データのスタイルの色の設定部分。
3	conditions:	分類する条件と配置するの色の設定部分。分類する条件と色を配列型で記述する。 例題では［分類する条件,配色する色］の組合せが配列型で格納されている。
4	例題における条件 １つ目	["Number(${地上階数}) >= 4","rgba(255, 0, 0, 0.5)"] ⇒ "Number(${地上階数}) >= 4"部分が分類の条件を表し、"rgba(255, 0, 0, 0.5)"部分が配色の色を設定している。 地上階数が４以上であれば合致し、赤に配色される設定。 ▼分類の条件に関して補足。 ${地上階数}で付与されている情報は、数字以外の情報も含まれているため、Number(${地上階数})と記述することで数値に変換している。 ${地上階数}で付与されている情報が数字以外の場合、数値変換後の値は異常となり、7行目で設定される色が適用される。 ５行目、６行目も同様。 ▼配色の色の設定に関して補足。 色の指定方法は rgba(red,green,blue,alpha)形式で指定している。 alpha は透過度を表し、0～1 の小数値で指定する。 0（0%）=透明、1（100%）=不透明となる。 ５行目、６行目も同様。
5	例題における条件 ２つ目	["Number(${地上階数}) >= 3","rgba(0, 255, 0, 0.5)"] ⇒ "Number(${地上階数}) >= 3"部分が分類の条件を表し、"rgba(0, 255, 0, 0.5)"部分が配色の色を設定している。 ４行目に合致しないデータのうち、地上階数が３以上であれば合致し、緑に配色される設定。
6	例題における条件 ３つ目	["Number(${地上階数}) >= 2","rgba(0, 0, 255, 0.5)"] ⇒ "Number(${地上階数}) >= 2"部分が分類の条件を表し、"rgba(0, 0, 255, 0.5)"部分が配色の色を設定している。 ５行目に合致しないデータのうち、地上階数が２以上であれば合致し、青に配色される設定。
7	例題における条件 ４つ目	["true","rgb(255, 255, 255)"] ⇒ "true"部分が分類の条件を表し、"rgb(255, 255, 255)"部分が配色の色を設定している。 4～6 行目に合致しないデータに対し、白に配色される設定。

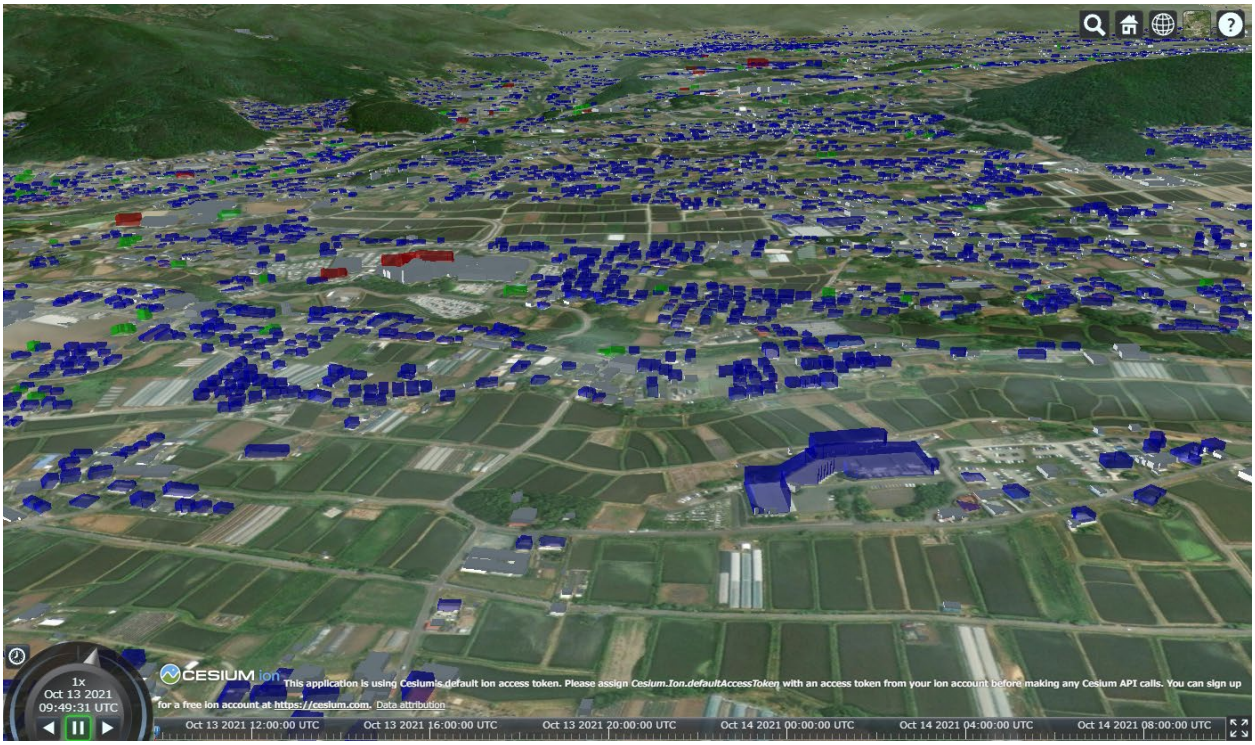
⑤ 本書２－１⑤と同様に、GitHub Desktop 経由で「chino_3dtiles3.html」をアップロードします。

⑥ 本書２－１⑥と同様に、アップロードした「chino_3dtiles3.html」を表示させます。

表示させた結果は下図のようになります。



マウスで視点を動かすと、下図のようになります。



本項目では建物の階数で分類し、色分けを行いましたが、付与されているその他の情報に対しても分類し、色分けすることが可能です。

巻末の『【補足資料２】物件の色分けにおける条件の記述について』では、分類するための条件の記述方法や主な付与情報の例を載せています。

興味のある方はぜひ試してみてください。

以上で本項目および本編は終了です。

【補足資料１】視点の緯度や経度、角度などの求め方

Cesium の初期視点を設定する際、表示したい場所の「緯度」や「経度」、「視点の高さ」、「横方向（東西南北の向き）」、「縦方向（天地の方向）」の値が必要になります。

そこで、Google chrome のディベロッパーツールを利用して、それぞれの値の求める方法について紹介します。

---

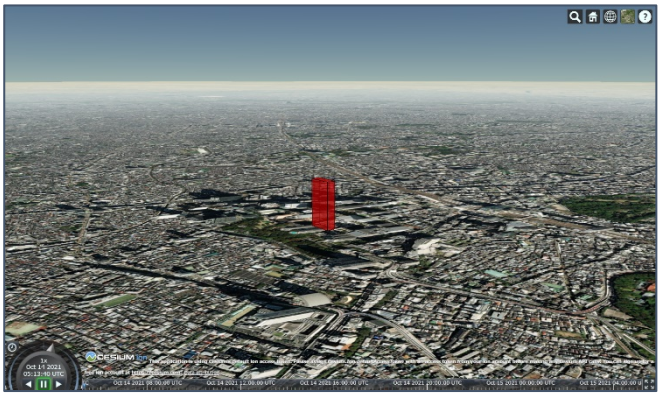
① Google chrome で閲覧させたいデータを表示します。

右図は例として、前頁の「HelloWorld7.html」を表示しています。

---

② マウスを操作し“表示したい場所”や“表示したい角度”で画面を表示します。

右図は例として、東京都庁を南西側から見る視点で視点の高さや角度を変えた画面を表示しています。



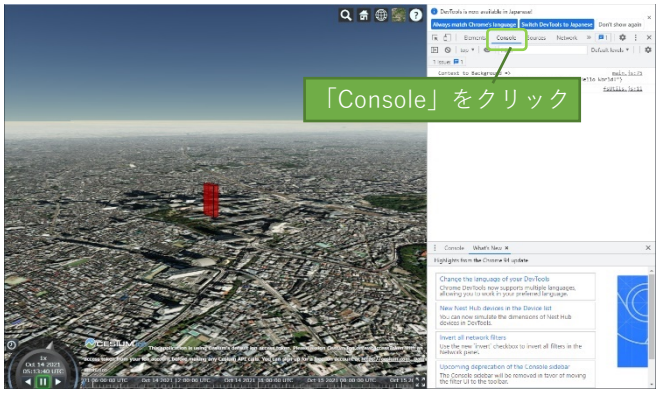
---

③ 画面右上の「Google chrome の設定」からディベロッパーツールを起動します。

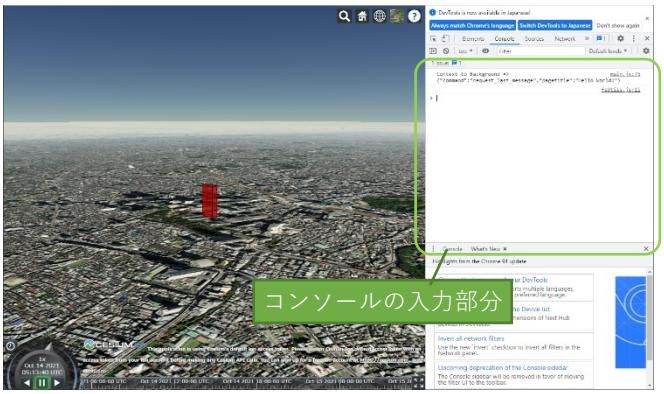
（1）「Google chrome の設定」をクリック。
（2）「その他のツール」をクリック
（3）「ディベロッパーツール」をクリック

ショートカットキーとして、キーボードから「Ctrl」＋「Shift」＋「I」を押下しても起動します。

④ ディベロッパーツールの「Console」をクリックしコンソール画面を表示します。



⑤ コンソールの入力部分に下記のコマンドを入力し Enter キーを押下すると、画面で表示している条件の「経度」「緯度」「視点の高さ」「横方向（東西南北の向き）」、「縦方向（天地の方向）」の値を取得できます。



「経度」を求めるコマンド

```
Cesium.Math.toDegrees(viewer.camera.positionCartographic.longitude)
```

「緯度」を求める場合コマンド

```
Cesium.Math.toDegrees(viewer.camera.positionCartographic.latitude)
```

「視点の高さ」を求めるコマンド

```
viewer.camera.positionCartographic.height
```

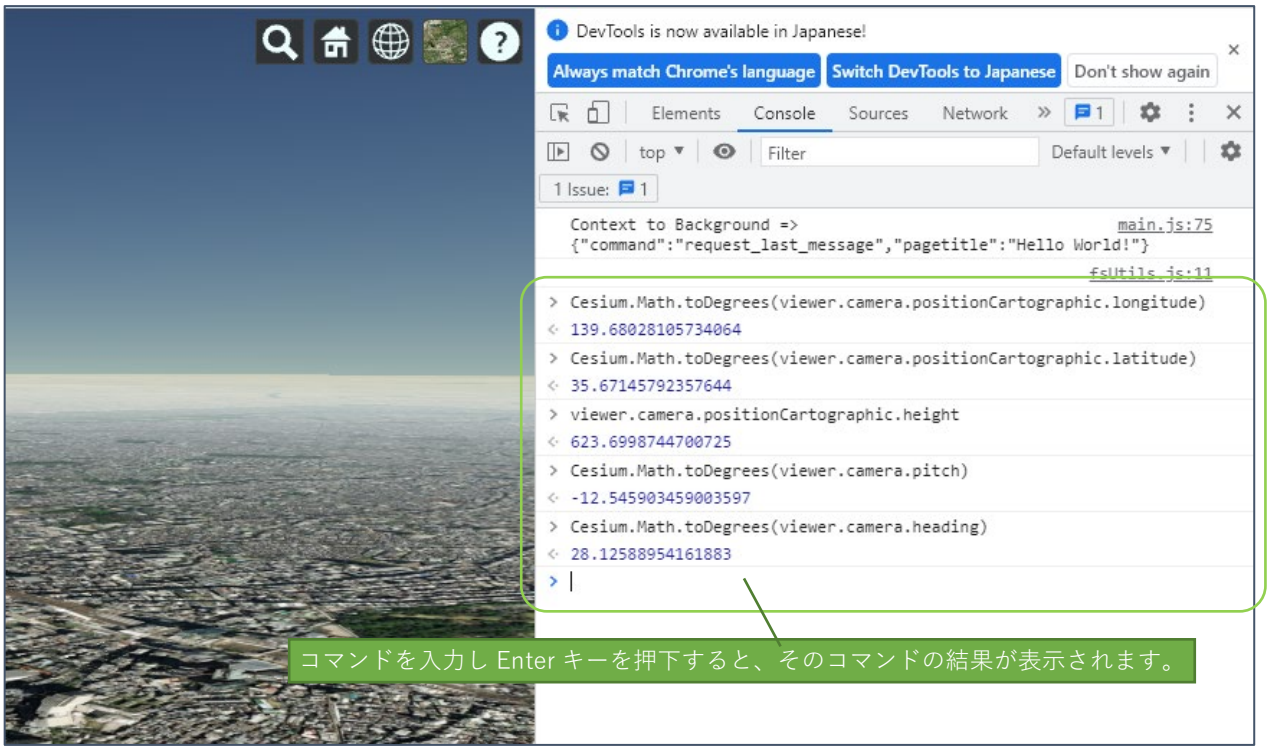
「縦方向（天地の向き）」を求めるコマンド

```
Cesium.Math.toDegrees(viewer.camera.pitch)
```

「横方向（東西南北の向き）」を求めるコマンド

```
Cesium.Math.toDegrees(viewer.camera.heading)
```

実際にコマンドを入力し Enter キーを押下した際のコンソール入力画面は下図のようになります。



【値を利用する際のワンポイント】

ここで得られる値は小数点以下が 10 桁以上の細かい数値になっています。

初期視点の設定に用いる場合、「横方向（東西南北の向き）」や「縦方向（天地の方向）」、「視点の高さ」に関しては、整数部分または小数点以下 1 桁程度の値に丸めた値でも差し支えありません。

また、「経度」「緯度」に関しても小数点以下 4 桁程度の値に丸めても大きな影響はありませんが、厳格に「経度」「緯度」を指定したい場合は、得られた値をそのまま設定してください。

以上で緯度や経度、角度などの値の求め方紹介を終了します。

【補足資料２】空間座標の「緯度」と「経度」の求め方

Cesium 上にポリゴンを配置する場合、ポリゴンの角を示す点群（以下、ポイント群）の空間座標（主に「緯度」と「経度」）が必要になります。

そこで、ここでは国土地理院が提供する地理院地図を用いた Web サービスを活用し、ポリゴンを配置したい地点の空間座標（「緯度」と「経度」）を求める手順について説明します。

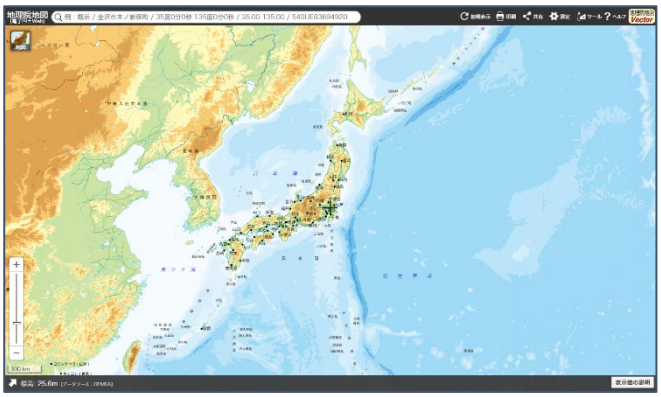
事例として、本書「２－２．レイヤの追加」で取得する東京都庁のポイント群の「緯度」と「経度」を求めます。

---

① 国土地理院が提供する Web サービス「地理院地図 / GSI Maps」にアクセスします。

・地理院地図 / GSI Maps
https://maps.gsi.go.jp/

右図は上記サイトにアクセスした際の初期画面です。



---

② 東京都庁にズームアップします。

画面左上にある検索窓から「東京都庁」を検索すると容易に検索でき、便利です。

また、ズームアップはマウスのホイール操作または画面左下のスケールバーで操作します。

③「作図･ファイル」の操作パネルを表示します。

（1）左上の『ツール』をクリック。
（2）右側に表示されるメニューから
『作図・ファイル』をクリック。

「作図・ツール」の操作パネルが表示されます。

④「面（多角形）を追加」をクリックします。

⑤ 地図上をクリックし、東京都庁を示す 4 点を結びます。

（1）開始位置（1 点目）をクリックします。

（2）２点目、３点目をクリックします。

クリックしていくと、囲まれた部分が網掛けで表示されます。
また、右クリックすると一つ前に戻ります。

（3）最後の点（4 点目）は**ダブルクリック**します。

⑥ 表示されたパネルの『OK』をクリックします。

なお、「緯度」「経度」の情報を取得することが目的であるため、表示されたパネルへの情報の追加入力や設定の変更は必要ありません。

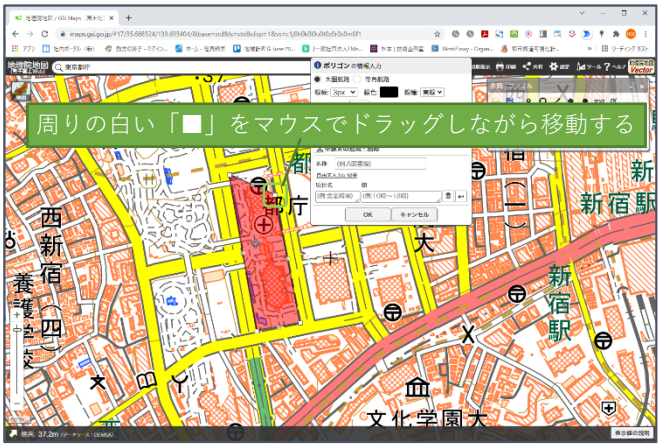
⑦ 4 点で選択された範囲が決まります。

選択範囲を編集する場合は“ペン”のアイコンをクリックします。また、削除する場合は“ゴミ箱”のアイコンをクリックします。

補足：選択範囲の編集方法

“ペン”のアイコンをクリックすると、本章⑥の画面に戻ります。

選択範囲の周りの白い「□」をマウスでドラッグしならが移動することで、選択範囲を自由に変更することができます。



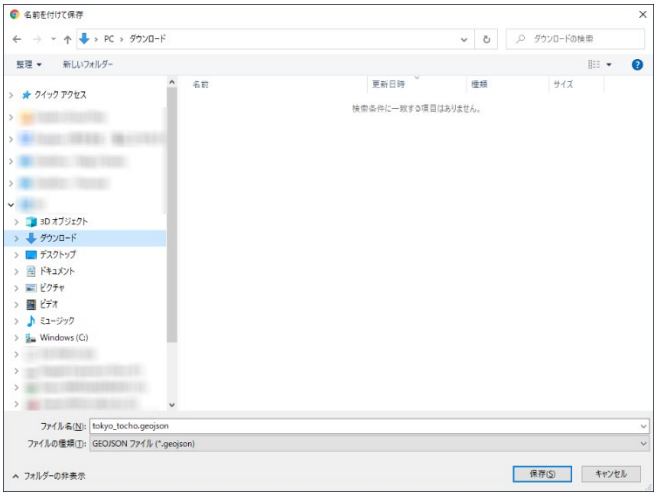
⑧ 「作図・ツール」の操作パネルの【確定】ボタンをクリックし確定します。

⑨ 「作図・ツール」の操作パネルの【保存】アイコンをクリックし確定します。

⑩ ファイル形式の選択で（１）『GeoJSON 形式』を選択し、（２）【上記の内容で保存】をクリックします。

⑪ （１）保存するファイル名を入力し、
（２）【保存】をクリックします。

⑫ 保存先を選択し、ファイルを保存します。

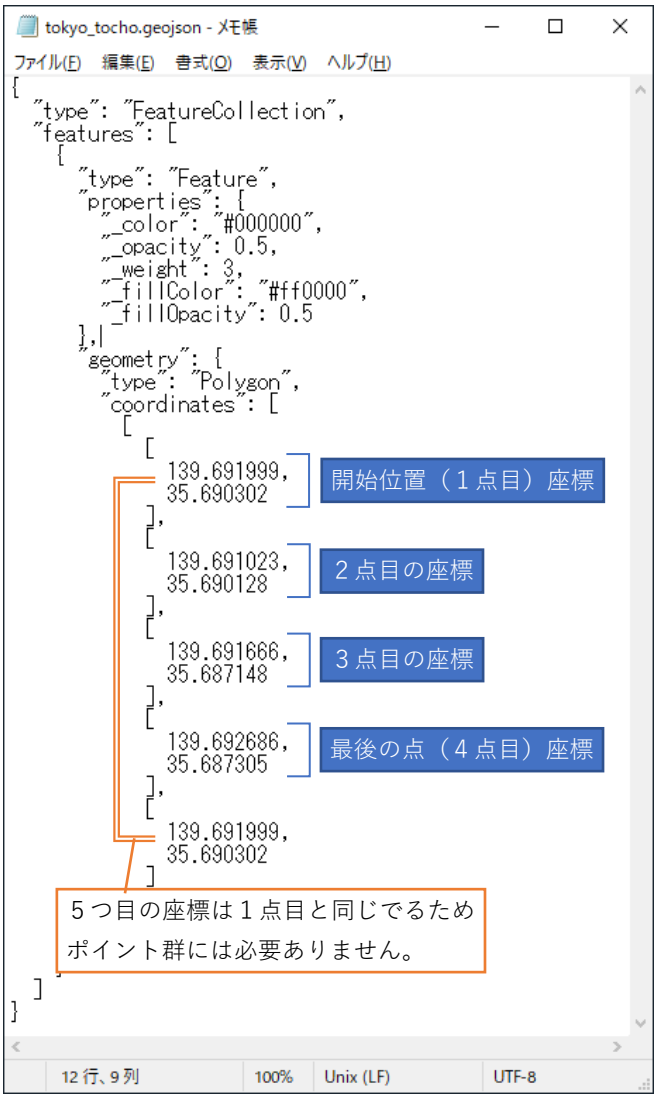


⑬ 保存したファイルをメモ帳で開きます。

“coordinates”で記述されている部分が選択した範囲の空間座標（緯度、経度）を示しています。

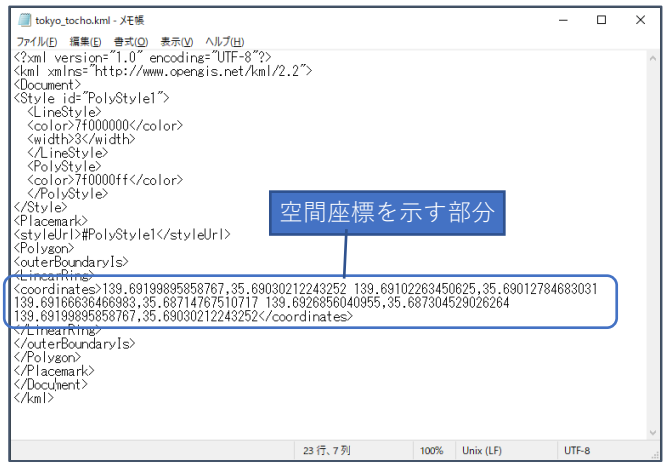
記述されている座標は５点分あります。
上から順に
・開始位置（１点目）のクリック地点
・２点目のクリック地点
・３点目のクリック地点
・最後の点（4 点目）のダブルクリック地点
を示しており、５点目は、範囲を閉じるために、開始位置（１点目）と同じ座標になっています。

従って、ポイント群としては、**開始位置（１点目）から最後の点（４点目）**の空間座標（緯度、経度）を用います。



補足：KML データで保存したデータ

本章⑩で KML データを選択し保存した場合、その KML データをメモ帳で開くと右図のようになります。
GeoJSON データと同様に“coordinates”で記述されている部分が選択した範囲の空間座標（緯度、経度）を示しており、GeoJSON データよりも細かい座標で取得できます。
しかし、各点の区切りが分かりにくい表記になっているため、GeoJSON データで取得することをお勧めします。



以上で空間座標の「緯度」と「経度」の求め方の説明を終了します。

【補足資料３】色の設定部分の記述方法の説明

「２－２．レイヤの追加」で説明した色の設定部分を例に説明します。
色の設定の記述方法は複数ありますが、主に下記の２つの記述方法を用います。

No	記述例
記述例１	Cesium.Color.RED.withAlpha(0.5)
記述例２	Cesium.Color.fromBytes(255, 0, 0,127)

記述例１は、予め Cesium で設定されている“色”を指名し、withAlpha()部分で透過を設定する記述方法です。
記述例２は RGB（赤、緑、青）および透過の設定を 0～255 の値で設定する記述方法です。
上記で示すこの２つの記述方法は同じ意味になります。

「２－２．レイヤの追加」では、「material:」で設定するポリゴンの色、および「outlineColor:」で設定するポリゴンの“辺”の色のいずれも記述例１の記述方法に則して記述されています。

（２－２．レイアの追加より抜粋）

```
material：Cesium.Color.RED.withAlpha(0.5),
・・・
outlineColor：Cesium.Color.BLACK
```

但し、「outlineColor:」で記述されている『Cesium.Color.BLACK』は、記述例１の withAlpha()部分を除いた記述方法になり、透過の設定をしない場合の記述方法です。
withAlpha()部分を withAlpha(1)とした場合は“透過しない事”を意味し“不透明”になり、withAlpha(0)の場合は“透明”となります。つまり、『Cesium.Color.BLACK』と『Cesium.Color.BLACK.withAlpha(1)』は同じ意味になります。
withAlpha()で設定する透過具合は 0～1 の間の小数値で設定します。下表に透過具合を変えた記述例を示します。

透過度	参考色	記述例1	記述例2
0		Cesium.Color.RED.withAlpha(0)	Cesium.Color.fromBytes(255,0,0,0)
0.25		Cesium.Color.RED.withAlpha(0.25)	Cesium.Color.fromBytes(255,0,0,63)
0.5		Cesium.Color.RED.withAlpha(0.5)	Cesium.Color.fromBytes(255,0,0,127)
0.75		Cesium.Color.RED.withAlpha(0.75)	Cesium.Color.fromBytes(255,0,0,191)
1		Cesium.Color.RED.withAlpha(1) または Cesium.Color.RED	Cesium.Color.fromBytes(255,0,0,255)

色を設定する場合、記述例１または記述例２のどちらを用いても問題ありませんが、細かい色味を指定したい場合は記述例２の記述方法で設定してください。

記述例１で設定する場合の代表的な色の記述例と、その同色を記述例２で設定する場合の記述例を下表に紹介します。

また、この表以外の色を設定したい場合は下記のリファレンス（英語）を参照してください。

・色に関するリファレンス（Cesium 公式ドキュメント（英語））

https://cesium.com/learn/cesiumjs/ref-doc/Color.html

色	記述例1	記述例2
黒	Cesium.Color.BLACK	Cesium.Color.fromBytes(0, 0, 0,255)
グレー	Cesium.Color.GRAY	Cesium.Color.fromBytes(127,127,127,255)
白	Cesium.Color.WHITE	Cesium.Color.fromBytes(255,255,255,255)
赤	Cesium.Color.RED	Cesium.Color.fromBytes(255, 0, 0,255)
緑	Cesium.Color.GREEN	Cesium.Color.fromBytes(0,255, 0,255)
青	Cesium.Color.BLUE	Cesium.Color.fromBytes(0, 0,255,255)
マゼンタ	Cesium.Color.MAGENTA	Cesium.Color.fromBytes(255, 0,255,255)
黄色	Cesium.Color.YELLOW	Cesium.Color.fromBytes(255,255, 0,255)
水色	Cesium.Color.CYAN	Cesium.Color.fromBytes(0,255,255,255)
オレンジ	Cesium.Color.ORANGE	Cesium.Color.fromBytes(255,165, 0,255)
ムラサキ	Cesium.Color.PURPLE	Cesium.Color.fromBytes(128, 0,128,255)
青緑	Cesium.Color.SPRINGGREEN	Cesium.Color.fromBytes(0,255,127,255)

【補足資料４】物件の色分けにおける条件の記述について

３－４では 3D Tiles データに付与されている建物の階数を、その階数を条件により色分けを行いました。その際に紹介した条件の記述方法について、少し掘り下げて説明します。

紹介する条件の記述方法は「JavaScript」言語に準拠しており、ここでは条件の記述方法の例示として一部についてのみを紹介しています。

（１）条件式の記述解説

３－４を例に、条件式の記述について解説します。３－４で用いた「chino_3dtiles3.html」の条件式の部分は下図の通りです。

chino_3dtiles3.html

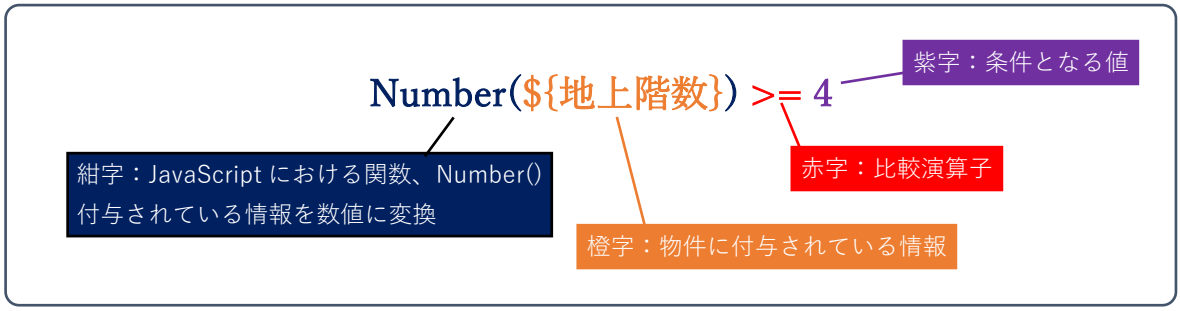
```
var tileset1 = new Cesium.Cesium3DTileset({ url: './sample/chino/tileset.json'});
tileset1.readyPromise.then(function(tileset1) {
  viewer.scene.primitives.add(tileset1);
  tileset1.style = new Cesium.Cesium3DTileStyle({
    color: {
      conditions:[
        ["Number(${地上階数}) >= 4","rgba(255, 0, 0, 0.5)"],
        ["Number(${地上階数}) >= 3","rgba(0, 255, 0,0.5)"],
        ["Number(${地上階数}) >= 2","rgba(0, 0, 255,0.5)"],
        ["true","rgb(255, 255, 255)"]
      ]
    },
    markerSymbol: '?'
  });
```

拡大＆着色

["Number(${地上階数}) >= 4","rgba(255, 0, 0,0.7)"],

この内、緑字部分が条件式になります。なお、**黒字**部分は conditions:で宣言される型の様式に沿った記述部分であり、青字部分は conditions:の型に沿って設定される色の記述部分です。**黒字**部分、青字部分の説明は割愛します。

この緑字部分をさらに分解し解説します。



橙字部分に条件としたい付与情報を記述します。（主な付与情報の一覧については後述を参照）
記述形式は、付与情報の名称を半角波括弧「{}」で括り、頭に半角ドルマーク「$」を設けた形式です。
赤字部分の比較演算子で左辺と右辺（紫字）を比較し、真偽を判定しています。
紺字部分は JavaScript における関数です。付与情報を“数値”に変換しています。（※）

まとめると、上記の場合、付与されている${地上階数}の情報を Number()関数で数値に変換した値（左辺）と、条件となる値「4」（右辺）とを比較演算子「>=」で比較し、真偽を判定しています。

※Number()関数で数値に変換している部分は、プログラムにおけるテクニックの一つです。
付与情報の内容によっては、数値として登録されているにも関わらず、プログラムの処理において“文字”と認識される場合があります。これは同じ項目の付与情報に空白や“null”文字、または全角で登録された数字文字など、“数値”として認識されない情報が含まれている場合があり、“全て数値である”と認識されないためです。
そこで、Number()関数を用いて“登録されている数字”を比較できる“数値”に変換しています。

（2）JavaScript における比較に用いる演算子

3－4 では、階数の分類に JavaScript における比較に用いる演算子を用いて、その物件の階数を判定し、分類（色分け）しています。

また、論理演算子を用いることで、複数条件による分類が可能になります。

JavaScript の代表的な比較演算子や論理演算子は下表の通りです。

比較演算子	例：A、B は値	解説
>	A > B	A が B よりも大きい場合、True。 それ以外は False。
>=	A >= B	A が B 以上の場合、True。 それ以外は False。
<	A < B	A が B よりも小さい場合、True。 それ以外は False。
<=	A <= B	A が B 以下の場合、True。 それ以外は False。
===	A === B	A が B と同じ場合、True。 それ以外は False。
!==	A !== B	A が B と異なる場合、True。 それ以外は False。

論理演算子	例	解説
||	条件式 X || 条件式 Y	|| は OR を意味する。 条件式 X または条件式 Y を満たす場合、True。
&&	条件式 X && 条件式 Y	&& は AND を意味する。 条件式 X かつ条件式 Y を満たす場合、True。

（3）条件式の記述例

${地上階数}のように付与されている情報が主に数字（数値）の場合、下表に示す記述例のように記述します。

No	比較演算子	記述例	解説
1	>	Number(${地上階数}) > 4	地上階数が 4 階より高い（5 階以上）物件を判断する場合に用いる。4 階以下の物件は対象外となる。
2	>=	Number(${地上階数}) >= 4	地上階数が 4 階以上の物件を判断する場合に用いる。4 階より低い物件は対象外となる。
3	<	Number(${地上階数}) < 4	地上階数が 4 階より低い（3 階以下）物件を判断する場合に用いる。4 階以上の物件は対象外となる。
4	<=	Number(${地上階数}) <= 4	地上階数が 4 階以下の物件を判断する場合に用いる。4 階より高い物件は対象外となる。
5	===	Number(${地上階数}) === 4	地上階数が 4 階である物件を判断する場合に用いる。4 階以外の物件は対象外となる。
6	!==	Number(${地上階数}) !== 4	地上階数が 4 階以外の物件を判断する場合に用いる。4 階である物件は対象外となる。

また、**${用途}**（主な付与情報の一つ。後述を参照）のように主に文字列とし付与されている場合は下表のように記述します。なお、文字列の比較においてはその大小を示す比較が単純ではないため、主に「===」や「!==」による比較を行います。

No	比較演算子	記述例	解説
1	===	${用途} === ‘住宅’	用途が住宅である物件を判断する場合に用いる。用途が住宅でない物件は対象外となる。
2	!==	${用途} !== ‘住宅’	用途が住宅でない物件を判断する場合に用いる。用途が住宅である物件は対象外となる。

文字列の比較の場合、左辺は付与情報を波括弧「{}」で括り、頭にドルマーク「$」を設けた形式にし、右辺は文字列をシングルクォーテーション「’」で括る形式にします。

（４）複数条件における記述例

複数条件で判断したい場合は論理演算子を用いて記述します。下記に記述例を示し解説します。

１）複数条件の記述例（緑字部分）　その１：

```
["(Number(${地上階数}) >= 4) && (${用途} === '住宅')","rgba(255, 0, 255,0.7)"]
```

「(Number(${地上階数}) >= 4)」と「(${用途} === '住宅')」を、AND 条件を表す「&&」で繋げ、複数条件を設定しています。

上記の場合、地上階数が「4」階以上である物件かつ用途が「住宅」である物件を色付けするようになります。

２）複数条件の記述例（緑字部分）　その２：

```
["(Number(${地上階数}) === 1) || (${用途} !== '住宅')","rgba( 0, 255, 255,0.7)"]
```

「(Number(${地上階数}) === 1)」と「(${用途} !== '住宅')」を、OR 条件を表す「||」で繋げ、複数条件を設定しています。

上記の場合、地上階数が「1」階である物件または用途が「住宅」ではない物件を色付けするようになります。

（5）主な付与情報の一覧

条件に用いる主な付与情報の例を下表に示します。

<table>
<tr><th>付与情報</th><th>記述例</th><th>解説</th></tr>
<tr><td>分類</td><td>${分類} === ‘AAA’
${分類} !== ‘AAA’</td><td>左辺側：${分類} と記述する。
右辺側：AAA には Building_class.xml で指定された項目（文字列）を記述し、その前後をシングルクォーテーション「’」で括る。
Building_class.xml で指定される項目（文字列）は以下の 5 種類。
<table>
<tr><th>No</th><th>AAA に記述する文字列</th></tr>
<tr><td>1</td><td>普通建物</td></tr>
<tr><td>2</td><td>堅ろう建物</td></tr>
<tr><td>3</td><td>普通無壁舎</td></tr>
<tr><td>4</td><td>堅ろう無壁舎</td></tr>
<tr><td>5</td><td>分類しない建物</td></tr>
</table>
</td></tr>
<tr><td>用途</td><td>${用途} === ‘AAA’
${用途} !== ‘AAA’</td><td>左辺側：${用途} と記述する。
右辺側：AAA には Building_usage.xml で指定された項目（文字列）を記述し、その前後をシングルクォーテーション「’」で括る。
Building_usage.xml で指定される項目（文字列）は以下の 18 種類。
<table>
<tr><th>No</th><th>AAA に記述する文字列</th></tr>
<tr><td>1</td><td>業務施設</td></tr>
<tr><td>2</td><td>商業施設</td></tr>
<tr><td>3</td><td>宿泊施設</td></tr>
<tr><td>4</td><td>商業系複合施設</td></tr>
<tr><td>5</td><td>住宅</td></tr>
<tr><td>6</td><td>共同住宅</td></tr>
<tr><td>7</td><td>店舗等併用住宅</td></tr>
<tr><td>8</td><td>店舗等併用共同住宅</td></tr>
<tr><td>9</td><td>作業所併用住宅</td></tr>
<tr><td>10</td><td>官公庁施設</td></tr>
<tr><td>11</td><td>文教厚生施設</td></tr>
<tr><td>12</td><td>運輸倉庫施設</td></tr>
<tr><td>13</td><td>工場</td></tr>
<tr><td>14</td><td>農林漁業用施設</td></tr>
<tr><td>15</td><td>供給処理施設</td></tr>
<tr><td>16</td><td>防衛施設</td></tr>
<tr><td>17</td><td>その他</td></tr>
<tr><td>18</td><td>不明</td></tr>
</table>
</td></tr>
</table>

付与情報	記述例	解説
建築年	Number(${建築年}) >= B Number(${建築年}) < B Number(${建築年}) === B ほか	左辺側：Number(${築年数}) と記述する。 右辺側：B は正の整数値。半角数字で記述する。 築年数が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。
地上階数	Number(${地上階数}) >= B Number(${地上階数}) < B Number(${地上階数}) === B ほか	左辺側：Number(${地上階数}) と記述する。 右辺側：B は正の整数値。半角数字で記述する。 地上階数が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。
地下階数	Number(${地下階数}) >= B Number(${地下階数}) < B Number(${地下階数}) === B ほか	左辺側：Number(${地下階数}) と記述する。 右辺側：B は正の整数値。半角数字で記述する。地下 1 階であれば「1」と記述する。符号は付加しない。 地下階数が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。
敷地面積	Number(${建物利用現況_敷地面積}) >= B Number(${建物利用現況_敷地面積}) < B Number(${建物利用現況_敷地面積}) === B ほか	基本属性である「建物利用現況」の拡張属性。 左辺側：Number(${建物利用現況_敷地面積}) と記述する。 右辺側：B は正数値。半角数字で記述する。 敷地面積が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。
延床面積	Number(${建物利用現況_延床面積}) >= B Number(${建物利用現況_延床面積}) < B Number(${建物利用現況_延床面積}) === B ほか	基本属性である「建物利用現況」の拡張属性。 左辺側：Number(${建物利用現況_延床面積}) と記述する。 右辺側：B は正数値。半角数字で記述する。 延床面積が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。
建築面積	Number(${建物利用現況_建築面積}) >= B Number(${建物利用現況_建築面積}) < B Number(${建物利用現況_建築面積}) === B ほか	基本属性である「建物利用現況」の拡張属性。 左辺側：Number(${建物利用現況_建築面積}) と記述する。 右辺側：B は正数値。半角数字で記述する。 建築面積が登録されていない物件も存在するため、Number()関数を用いて、数字を数値に変換する必要がある。

<table>
<tr><th>付与情報</th><th>記述例</th><th>解説</th></tr>
<tr><td>耐火構造種別</td><td>${建物利用現況_耐火構造種別} === ‘AAA’
${建物利用現況_耐火構造種別} !== ‘AAA’</td><td>基本属性である「建物利用現況」の拡張属性。
左辺側：${建物利用現況_耐火構造種別} と記述する。
右辺側：AAA には Building_fireproofStructureTpye.xml で指定された項目（文字列）を記述し、その前後をシングルクォーテーション「’」で括る。
Building_fireproofStructureTpye.xml で指定される項目（文字列）は以下の 4 種類。
<table>
<tr><th>No</th><th>AAA に記述する文字列</th></tr>
<tr><td>1</td><td>耐火</td></tr>
<tr><td>2</td><td>準耐火造</td></tr>
<tr><td>3</td><td>その他</td></tr>
<tr><td>4</td><td>不明</td></tr>
</table>
</td></tr>
<tr><td>集客施設立地状況_分類</td><td>${集客施設立地状況_分類} === ‘AAA’
${集客施設立地状況_分類} !== ‘AAA’</td><td>基本属性である「集客施設立地状況」の拡張属性。
左辺側：$集客施設立地状況_分類} と記述する。
右辺側：AAA には LargeCustomerFacilities_class.xml で指定された項目（文字列）を記述し、その前後をシングルクォーテーション「’」で括る。
LargeCustomerFacilities_class.xml で指定される項目（文字列）は以下の 5 種類。
<table>
<tr><th>No</th><th>AAA に記述する文字列</th></tr>
<tr><td>1</td><td>大規模小売店舗（食品スーパー）</td></tr>
<tr><td>2</td><td>大規模小売店舗（百貨店・スーパー・ショッピングセンター・寄合百貨店・小売市場）</td></tr>
<tr><td>3</td><td>大規模小売店舗（ホームセンター・専門店（家具・家電・書籍等））</td></tr>
<tr><td>4</td><td>大規模小売店舗（その他）</td></tr>
<tr><td>5</td><td>大規模集客施設（ 床面積１万㎡超の店舗、映画館、アミューズメント施設、展示場等）</td></tr>
</table>
</td></tr>
</table>

以上で物件の色分けにおける条件の記述についての説明を終了します。