Turi Zoltán

Neptun:G4R8AJ

Cesium JS dokumentáció

Felhasználói Dokumentáció

Előfeltételek:

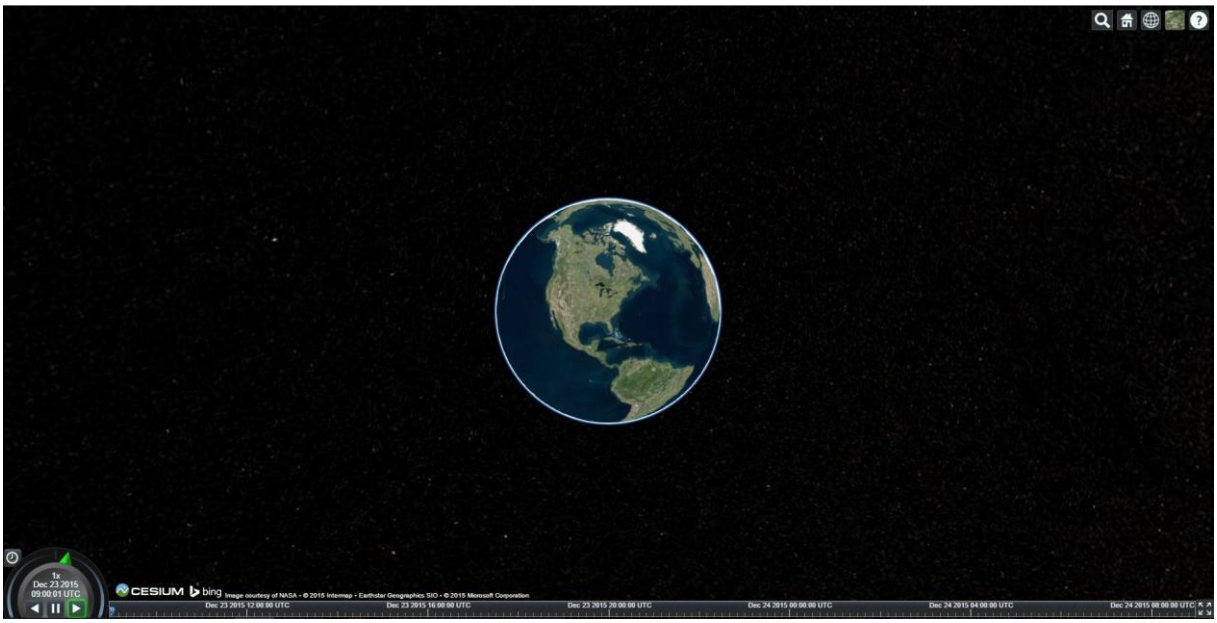
- Node.js és Node Package Manager: nodejs.org/en/download/stable linkről telepíthető mindkettő

Telepítés:

- Letöltjük az archívumot az cesiumjs.org/downloads.html linkről és kicsomagoljuk egy mappába
- Nyitunk egy parancssort és benavigálunk abba a mappába, ahová kicsomagoltuk(cd)
- kiadjuk az npm install parancsot
- amikor ez végzett készen állunk a saját gépi használatra(node server.js). (server esetén ez azt jelenti, hogy az archívumot egy olyan serverre csomagoljuk ki, ahol node.js fut)

Használat

Az alap földgömb megjelenítéséhez a Build/Cesium, vagy a Source mappák Cesium.js javasrcipt fájllát és a Build/Cesium/Widgets, vagy a Sourve/Widgets widgets.css-ét kell behúzni



További példákat lehet elérni az elindított serveren(localhost:8080 – node server.js paranccsal és egy sandboxot, amin lehet kísérletezni)

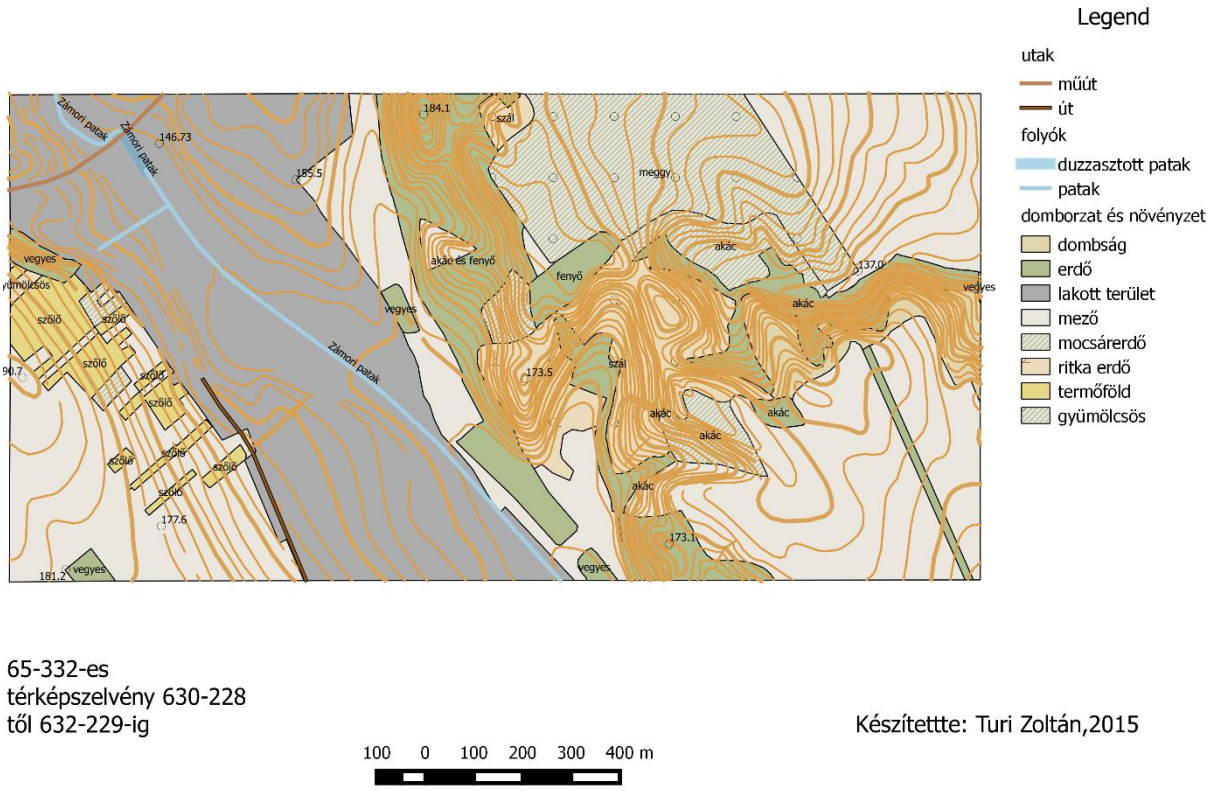
A következőkben megpróbálom a második térinformatika beadandót ebbe a rendszerbe beimportálni és ennek sikerességét és nehézségeit fogom felmérni.

A feladat leírása és megoldása

A feladat leírása:

Adatok importálása GeoJson állományokból, azok egyedi stílusozása és ezekre zoomolása a kamerának. Illetve ezen adatok összehasonlítása a terepi/jelenlegi adatokkal.

1. A GeoJson/TopoJson importálás lehetőségét, minőségét, képességeit teszteljük
2. A kamerakezelés lehetőségeit teszteljük le
3. A stílusozás lehetőségeit(a beimportált állományok mennyire stílusozhatóak)



65-332-es
térképszelvény 630-228
től 632-229-ig

Készítettte: Turi Zoltán,2015

A cél a fenti térképhez hasonló(a szövegek és jelmagyarázat nélküli) kinézet előállítása a vonalas és területes ábrázolásra nézve.

A Feladat Megoldása(Vektoros adatok megjelenítése)

A projekt a localhost:8080 ról a georeferencing test app linken érhető el, miután elindítottuk a lokális szervert(Beadandó könyvtárában kiadjuk a node server.js parancsot)

Adatok georeferálása: ezt a feladatot a Cesium.js nem tudja elvégezni, de egy kapott GeoJson, vagy TopoJson állományt már fel tud dolgozni, erre egy külső szoftver kell, mint például a Qgis

Adatok importálása Jsonból(Geo, vagy TopoJson):

A viewer objektumunknak kell odaadni a betöltött dataSource-t a viewer.dataSources.add funkció segítségével

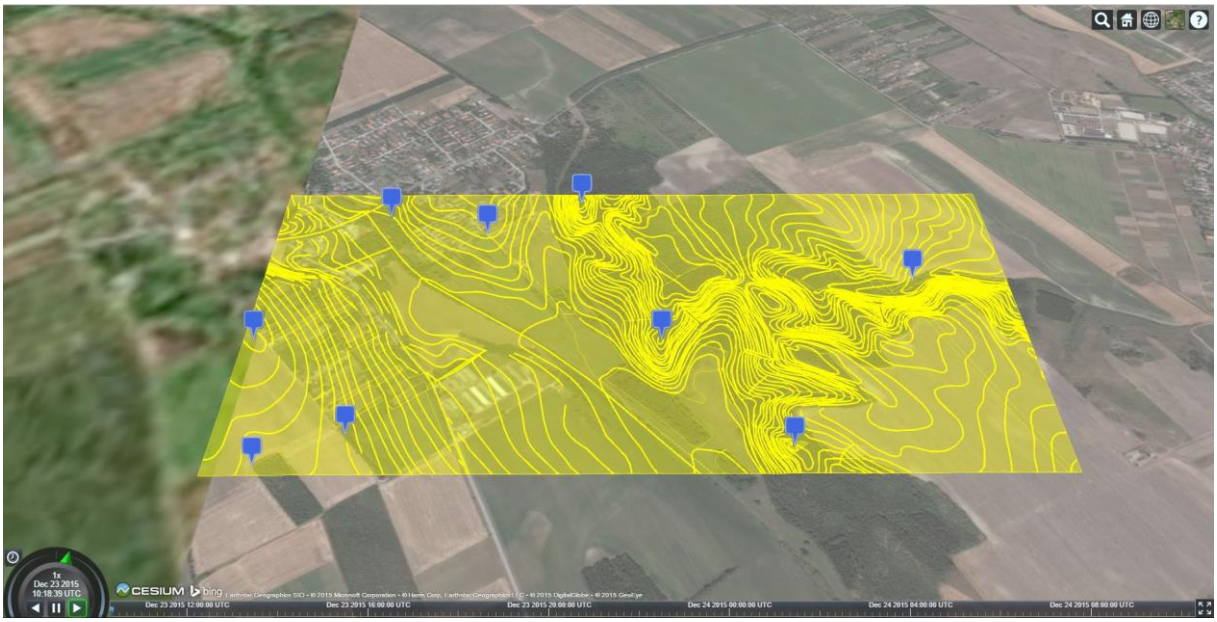
A json beolvasását a Cesium.GeoJsonDataSource.load funkció végzi el nekünk, így megkapjuk az importált adatokat a térképen. A funkció visszatérési értéke a betöltött adatokat tartalmazza megfelelő struktúrában

Ebből az objektumból az entitásokat a dataSource.entities-el kérhetjük el, azok adatait pedig a dataSource.entities.values

Az entitások eredeti attribútumai a dataSource.entities.values-ből előálló tömb minden elemének a property attribútuma.

A szükséges vetület a(z) WSG 84(EPSG 4326)-es

Ha mindent jól csináltunk egy ilyen térképet kellett, hogy kapjunk:



Amint látjuk ez még nehezen felismerhető, de a következőkben orvosoljuk ezt a problémát is

Az importált adatokból a CesiumJs keretrendszer entitásokat készít, amelyeket stílusozhatunk is, ha akarunk(álatában akarunk), amelyet a következő pontban tárgyalunk.

Entitások stílusbeállításai

A betöltött entitások stílusát a dataSource.then(function(dataSource){}) alatt lehet beállítani. Ez biztosítja, hogy a betöltés után fog a meghívott függvényünk lefutni.

Minden entitásr külön állíthatjuk be a stílust, ha véletlenszerű szinezést szeretnénk, akkor Cesium.Color.fromRandom({alpha: 1.0}) kóddal megadhatjuk azt.

A poligon és a polyline material attribútuma állítja annak színét. Az outline a polygonok/polylineok körülrajzolását állítja, falsera állítva azt elhagyhatjuk. Az outlinenak beállítva a color és a width attribútumát pedig a körülrajzolást szabályozhatjuk

A pontok stílusozása is hasonlóan történik, csak:

- A pontoknak nincs material attribútuma, ehelyett colorja van
- A pontok méretét a pixelSize állítja be, alapból 1

Egy-egy entitás elhelyezkedését a position adat határozza meg

Ha mindez megvan a térképünk így fog kinézni(A magasságpontok,szintvonalak,utak és folyók színe fix, a területek színe véletlenszerű):

Amint látjuk a pontok kirajzolása nehézkes/átfed és több helyen a terep és a vonalak kirajzolása is egymásba átfed – ez a georeferálás Z koordinátájának hiánya miatt van – így a rendszer mindent egy magasságúnak vett.

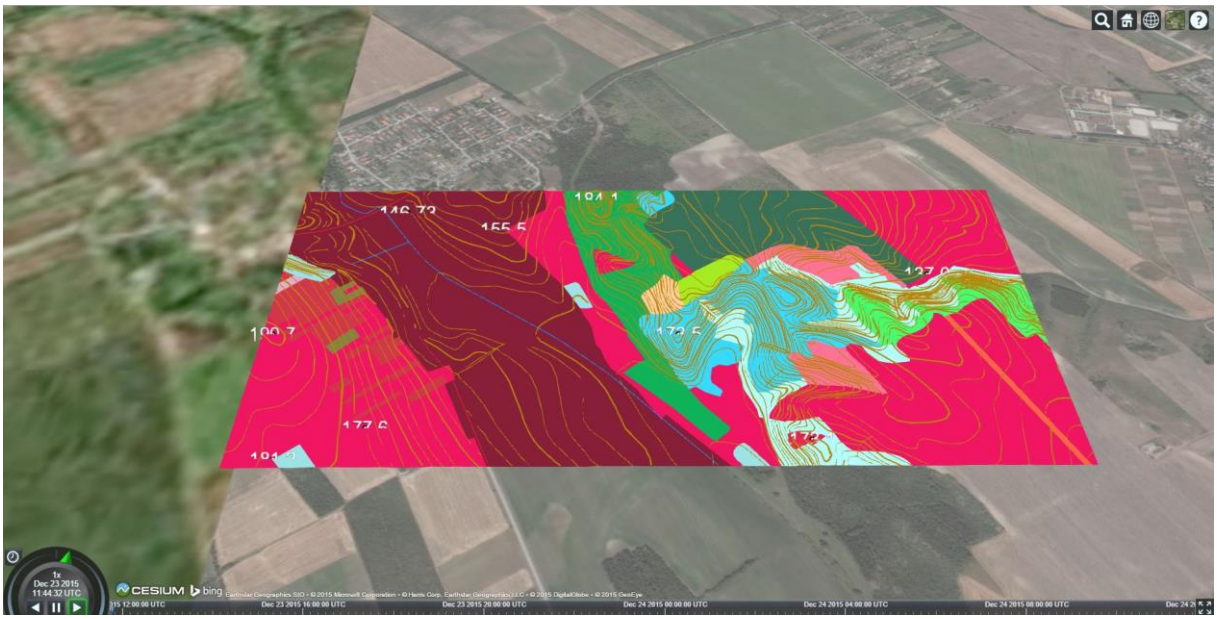
Értelemszerűen, ha 3d-s rendszerbe georeferálunk, akkor a Z koordinátákat is be kell állítanunk a rendszerben minden alakzatra/pontra és azt kell exportálni, majd importálni

Szövegezés

Az entitások szövegezését a magasságpontok magasságának megadásával fogjuk bemutatni. A jelenlegi objektum label objektuma adja meg a kíírandó szöveget:

- A text attribútuma az objektumnak a szöveget adja meg
- A style attribútuma az objektumnak a stílust(kitöltés, kitöltés és körülírás, csak körülírás)
- Lehet a szöveg karaktertípusát állítani
- A szöveg elhelyezkedését

A szövegezést a térképbe téve a következőt kapjuk:



Amint látjuk a pozicionálás alapbeállításban hibás(ugyan úgy a Z koordináták hiánya miatt, a pont magasságán „rajzolja a ki a szöveget is")

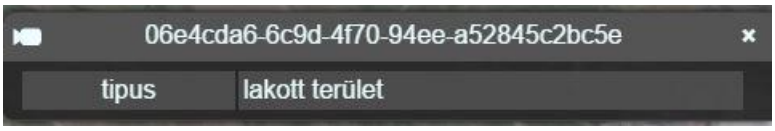
Ezen a pixelOffset állításával, azaz a szöveg átpozicionálásával segíthetünk: pixelOffset : new Cesium.Cartesian2(0, -12), ezzel a következő lesz a kinézet:

Amint láthatjuk így a szövegeket olvashatóvá tehetjük.

Egyéb érdekességek

Képeket, raszteres adatokat Vektoros adatok „grafikája”-ként tudunk importálni, mégpedig image material-ként, azaz szín helyett egy képre mutató hivatkozást adunk meg.

Megadhatunk teljes leírást, amit dinamikusan a térképről lekérhetünk az entitások adataival együtt:



Elég az egérrel az adott területre kattintanunk.

Entitások csoportjaira és entitásokra zoomolhatunk a viewer.zoomTo funkcióval, amelynek a paramétere a „célkeresztbe” vivendő entitás

Előnyök:

- Könnyen elsajátítható, a letöltött állománnyal egyben több példaalkalmazást és egy tesztkörnyezetet is kapunk még 10-12 példával
- Az adatok olvasása és kezelése egyszerű, mint ahogy a megjelenítésükhöz is csak egy HTML5-ös böngésző szükséges
- Könnyen telepíthető csomagban jön(egy npm install és a szükséges dolgokat letölti magának)
- Kiterjedt dokumentáció, amely szinte minden funkciót leír
- Gyors megjelenítés és könnyű kamerakezelés
- Különböző típusú földtérképeket is tud kezelni a megfelelő szolgáltató beállításával(WorldStreetMap, ESRI arcGIS mapServer, NASA Black Marble Imagery, stb..)
- A hibaüzenetek tisztán érthetőek(már amikor vannak)

Hátrányok:

- Magas erőforrásigény kliens oldalon
- Pontok kezelése nehézkes, nehéz őket mozgatni(pl 3d adatok hiánya miatt megemelni)
- Magas adatforgalom
- Node.js web server kell a futtatásához és node package manager(npm) (esetünkben a dev-web server látja el ezt a feladatot: node server.js)
- Hibákra érzékeny, hibás adatok esetén lehetséges, hogy nem ad hibaüzenetet, csak megáll a feldolgozás(nehezen hibakövethető)