SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1038

VIZUALIZACIJA VIŠESLOJNIH GEOINFORMACIJSKIH PODATAKA UZ POMOĆ WEB GRAFIČKOG POGONA CESIUM 3D

Ana Marija Komar

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 2. ožujka 2015.

Predmet: **Diplomski rad**

DIPLOMSKI ZADATAK br. 1038

Pristupnik: **Ana Marija Komar (0036456370)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Vizualizacija višeslojnih geoinformacijskih podataka uz pomoć Web grafičkog pogona Cesium 3D**

Opis zadatka:

Proučiti i ukratko opisati arhitekturu Web grafičkog pogona Cesium 3D i njezine mogućnosti i ograničenja. Implementirati 3D vizualizaciju za višeslojni geoinformacijski sustav u kojemu je svaki sloj predstavljen 3D geometrijom. Vizualizacija treba biti implementirana na plohi i na sferi. Slojevi se mogu po volji uključivati i isključivati pri čemu se mora osigurati konzistentnost i ispravna međusobna povezanost (bez šupljina i preklapanja između slojeva; statička ili dinamička) i podudaranje geometrija slojeva na temelju informacija o poziciji. Implementirati alat za odabir dijela geometrije jednog ili više slojeva.
Analizirati prednosti i nedostatke Web grafičkog pogona Cesium 3D za navedenu primjenu te predložiti poboljšanja u svrhu izgradnje dodatnih alata za manipulaciju i analizu 3D geometrija geoinformacijskih podataka. Dokumentirati sve korištene programske biblioteke, matematičke koncepte i algoritme te navesti korištenu literaturu.

Zadatak uručen pristupniku: 2. ožujka 2015.
Rok za predaju rada: 30. lipnja 2015.

Mentor:

Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Dielovođa:

Doc. dr. sc. Tomislav Hrkać

SADRŽAJ

1. UVOD

Sve brži razvoj računalne grafike, povećanje računalnih kapaciteta te novosti u web tehnologijama u potpunosti su unaprijedili tehnike vizualizacije. Najnoviji trendovi u vizualizacijama i animacijama 3D podataka promijenili su način na koji se informacije na računalu prikazuju. 3D vizualizacija stoga postaje sve češće korištena u mnogim područjima, a veliku korist od 3D vizualizacije podataka imaju geoinformacijski sustavi.

Danas su postupci pribavljanja 3D podataka relativno jeftini, a sam prikaz podataka u 3D pruža puno više informacija korisnih za analizu. 3D vizualizacija geoinformacijskih podataka se sastoji od slojeva. Svaki sloj je predstavljen jednom geometrijom, a da bi vizualizacija više slojeva bila uspješna, slojeve treba ispravno povezati. Podaci se najčešće prikazuju na plohi i na sferi.

Uvođenjem HTML5 standarda postalo je moguće unutar web preglednika (engl. *web browser*) prikazivati vrlo složene grafičke modele bez potrebe za instaliranjem dodataka. S obzirom na sve raširenu uporabu geoprostornih podataka pojavila se i potreba za razvojem alata koji će omogućiti lakši razvoj i prikaz karata i ostalih podataka unutar web preglednika.

U ovom radu se za implementaciju prikaza geoinformacijskih podataka koristi web grafički pogon (engl. *web engine*) Cesium. Cesium je JavaScript biblioteka za stvaranje i prikaz geoinformacijskih podataka poput 3D sfere ili 2D plohe unutar web preglednika. S obzirom da za prikaz podataka Cesium koristi WebGL, nisu potrebni nikakvi dodaci te se aplikacije razvijene u Cesiumu mogu pregledavati kroz različite preglednike.

Rad je podijeljen u šest poglavlja. Nakon prvog uvodnog, slijedi opis arhitekture Cesium 3D web grafičkog pogona. Treće poglavlje opisuje ostvarenje terena, višeslojnog geoinformacijskog sustava i razvijenih alata. Četvrto poglavlje daje pregled razvijenog sustava te načina njegova korištenja. U petom poglavlju navedeni su nedostaci i prijedlozi za poboljšanje Cesium-a. U šestom poglavlju iznesen je zaključak. Na kraju se nalazi dodatak koji opisuje što sve treba napraviti kako bi se razvijeni sustav uspješno pokrenuo na računalu.

2. CESIUM

Cesium je biblioteka otvorenog koda za razvoj klijentskog dijela arhitekture pri prikazu geoprostornih podataka na sferi i na plohi. Biblioteka je pisana u JavaScriptu, a koristi WebGL za grafički prikaz podataka. S obzirom da je razvijen isključivo za prikaz geoprostornih podataka, nudi višu razinu apstrakcije nego standardni web grafički pogoni. Na primjer, pogodniji je za prikaz različitih terena i slojeva, iscrtavanje vektorskih podataka iz standardnih formata, kontrolu kamere te precizno upravljanje prikazom podataka s veće udaljenosti ili prikazom velikih prostornih koordinata. S obzirom da je izgrađen na WebGL-u, sličan je uobičajenim grafičkim pogonima, pa tako i podržava prikaz 3D modela.

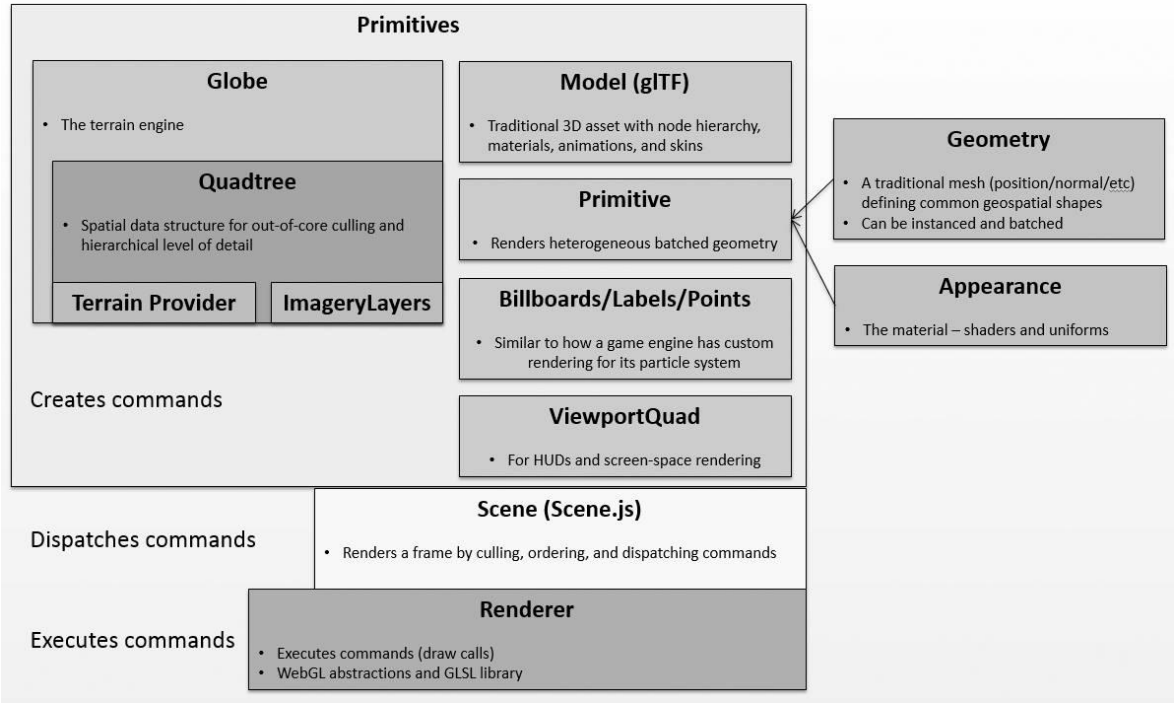
Cesium je naročito prikladan za dinamički prikaz geoprostornih podataka i to uz pomoć Cesium jezika (engl. *Cesium Language* - CZML). Može se integrirati sa slojevima koji se uvoze (engl. *import*) iz različitih izvora, npr. *ArcGIS Map poslužitelj*, *OpenStreetMap* itd. Također, moguće je uvesti vanjske WMS (*Web Map Service*) i TMS (*Tile Map Service*) slojeve. Pojedinim slojevima moguće je promijeniti svojstva poput, npr. svjetline, boje ili prozirnosti.

Podržane su biblioteke koje iscrtavaju 2D i 3D geometriju, pa tako korisnik može iscrtavati linije, poligone, elipsoide, sfere, labele, ploče, ali i složenije modele. Kontroliranje događaja s tipkovnice ili miša, pokreta kamere i zumiranje je već ugrađeno. Postoji mnoštvo ugrađenih materijala za opis vanjske površine objekata i slojeva, a mogu se uvesti i vanjski materijali. Također, podržane su matematičke biblioteke (*World Geodetic System, International Celestial Reference Frame*) koje omogućuju složenije matematičke izračune te se oslanjaju na postojeće standarde i dogovore o koordinatnim sustavima i kartezijevom sustavu.

Da bi se koristio Cesium nisu potrebni nikakvi dodaci za web preglednike jer Cesium koristi HTML5 standard koji je podržan u preglednicima. Može se pokrenuti na svim platformama koje koriste preglednike koji podržavaju WebGL, te uređajima s grafičkim karticama koje podržavaju WebGL. Moguće ga je iskoristiti i u drugim vrstama aplikacija koristeći web kontrole, npr. Microsoft-ova web kontrola preglednika.

2.1 Arhitektura Cesium-a

Arhitektura Cesium-a uključuje tri glavna sloja : sloj vizualizatora (engl. *Renderer*) , sloj scene (engl. *Scene*) i sloj primitiva (engl. *Primitive*) koji čine grafički stog. Izgled grafičkog stoga (engl. *graphical stack*) prikazuje Slika 2-1.



Slika 2-1 Grafički stog Cesiuma
(izvor: http://cesiumjs.org/2015/05/26/Graphics-Tech-in-Cesium-Stack/)

Najniži nivo grafičkog stoga je vizualizator. On predstavlja apstrakciju WebGL sloja koji upravlja WebGL resursima i izvršavanjem naredbe za iscrtavanje (engl. *Draw Command Execution*). Sloj scene izgrađen je nad slojem vizualizatora, a odgovoran je za iscrtavanje okvira zahtijevanih od strane viših slojeva Cesium-a. Scena obavlja odbacivanje okvira (engl. *culling*), uređuje njihov redoslijed prema određenom kriteriju te po potrebi dostavlja vizualizatoru. Najviši sloj stoga, građen na vizualizatoru i sceni, je sloj primitiva. On predstavlja stvarne objekte koji se prikazuju, a njima upravlja naredbama za stvaranje i šalje ih sceni.

2.1.1. Sloj vizualizatora

U sloju vizualizatora su centralizirani svi WebGL pozivi. Stoga, vizualizator pruža viši nivo apstrakcije i čini ostatak Cesium-a manje sklonim greškama.

Dio vizualizatora čini dinamički protočni sustav sjenčanja (engl. *shading pipeline*) te GLSL biblioteke koje sadrže konstante i funkcije. Također, dio pažnje usmjeren je na optimizaciju i ostvarivanje što boljih karakteristika prilikom iscrtavanja, pa je privremeno spremanje (engl. *caching*) i minimizacija broja WebGL poziva ostvarena na jednom mjestu. Obzirom da je WebGL stroj stanja vizualizator upravlja stanjima. Ovakva građa vizualizatora ga čini lako prenosivim i pogodnim za nadograđivanje.

Vizualizator je izgrađen nad slojem jezgre koja sadrži većinom matematičke funkcije niže razine koje se često koriste. Jezgra daje podršku za račun s matricama, vektorima, kvaternionima, omogućuje transformacije (npr. kartografski u kartezijev sustav), kartografske projekcije (npr. Merkatorova projekcija) te krivulje za interpolaciju položaja i orijentacije. Također, jezgra podržava pozicije sunca i Julijanski kalendar.

Primjer u kojem se koristi sloj jezgre je kada se želi podatke prebaciti iz jednog formata u drugi, kao kod promjene iz EPSG:4326 formata (WGS84 elipsoid) u kartezijev sustav, odnosno preračunavanja geografske duljine i širine u xyz koordinate.

2.1.1.1. Dijelovi vizualizatora

Objekti koji čine vizualizator su:

- *VertexArray* – sadrži skup atributa koji opisuju vrhove i opcionalni skup indeksa, a atributi i indeksi se nalaze u spremnicima
- *RenderState* – definira funkcijsko stanje grafičkog protočnog sustava za određeni poziv iscrtavanja
- *ShaderProgram* – predstavlja prevedeni program za sjenčanje koji radi izravno s Cesium-ovim matricama, kartezijanovim koordinatnim sustavom, bojama, teksturama i sfernim kartama
- *Framebuffer* – spremnik okvira sa svim potrebnim dijelovima koji čini osnova za poziv iscrtavanja

2.1.1.2.Protočni sustav sjenčanja

Iz koda koji opisuje obradu vrhova i fragmenata može se jednostavno kreirati program za sjenčanje jednom naredbom:

```
var shaderPrograms = context.getShaderCache().getShaderProgram(vs,fs);
```

Programi za sjenčanje smiju koristiti individualne .gsls datoteke, a Cesium ih prilagođuje prije daljnjeg prosljeđivanja. S obzirom da je većina programa za sjenčanje u Cesium-u dijelomično napisana, zasada je omogućeno izravno korištenje GLSL biblioteka unutar programa za sjenčanje koji se ne moraju dodatno uključivati, već ih se koristi kao ugrađene funkcije i prepoznaje po prefiksu *czm_* .

Na primjer, ako se želi izračunati interval presijecanja zrake i elipsoida, koriste se GLSL ugrađene funkcije:

```
v_position = (czm_modelView*position).xyz;
czm_ray ray = czm_ray(vec3(0,0), normalize(v_position);
czm_raySegment interval = czm_rayEllipsoidIntersectionInterval(ray,
                                                    ellipsoid;)
```

GLSL ugrađene funkcije mogu pozivati druge ugrađene funkcije čime stvaraju usmjereni aciklički graf (engl. *Directed Acyclic Graph* - DAG). Prilikom pokretanja, izvorni GLSL kod se predaje objektu *ShaderSource*, koji pronalazi *czm_* identifikatore i usmjerava DAG pri stvaranju konačnog izvora. Ovo se izvodi prilikom pokretanja kako bi se izbjeglo višestruko kopiranje istih GLSL ugrađenih funkcija, jer Cesium aplikacije mogu zahtijevati mnogo permutacija programa za sjenčanje koje se ne mogu odrediti prije pokretanja.

2.1.1.3.Izvršavanje naredbe crtanja

Prilikom poziva iscrtavanja potrebno je definirati sljedeće parametre: primitiv koji se iscrtava, programe za sjenčanje, konstante, listu vrhova i stanje vizualizatora. Koraci koji se obavaljaju su sljedeći:

- povezuje se spremnik okvira, ako je različit od prethodnog poziva
- ako je različito od prethodnog, primjenjuje se novo stanje vizualizatora
- dohvaća se, po potrebi i prevodi, program za sjenčanje
- povezuje se odgovarajući spremnik vrhova

Na kraju svakog okvira, čisti se stanje na način da se miče veza s programom za sjenčanje, spremnicima vrhova i tekstura. Ovo pomaže smanjenju količine stanja kojom vizualizator mora upravljati između izvođenja svake naredbe.

Stanje vizualizatora definira funkcijsko stanje grafičkog protočnog sustav za određeni poziv iscrtavanja te nema potrebe za brigom o globalnom stanju. Sljedeći primjer pokazuje definiranje stanja vizualizatora za uključen z-spremnik (provjera dubine), stražnje odsijecanje poligona i alfa miješanje boja te njegovo uključivanje pri pozivu iscrtavanja.

```
var rs = context.createRenderState({
            depthTest :  {
                    enabled : true
            },
            cull : {
                    enabled :true,
                    face : CullFace.BACK
            },
            blending : BlendingState.ALPHA_BLEND
});

context.draw({
            primitiveType : PrimitiveType.TRIANGLED,
            shaderProgram : sp,
            uniformMap : uniforms,
            vertexArray : va,
            renderState : rs
});
```

2.1.2. Sloj scene

Sloj scene ostvaruje većinu funkcionalnosti. On povrh sloja jezgre i sloja vizualizatora omogućuje plošne i sferne konstruktore visoke razine. Omogućuje 3 načina pogleda geoprostornih podataka unutar iste aplikacije: 3D sferu, 2D mapu te 2.5D *Columbus* pogled. Također, omogućuje prikaz slojeva visoke rezolucije iz različitih vanjskih izvora poput *ArcGIS MapServer*-a, *OpenStreetMap*-a i WMS-a (engl. *Web Map Service*). Osim složenijih geometrija moguće je iscrtavati i jednostavnije geometrije poput linija, poligona, elipsoida, oznaka i natpisa. Ovaj sloj, također, omogućuje upravljanje kamerom, te animacijama koje mijenjaju svojstva objekata tijekom vremena.

Scena predstavlja sve grafičke objekte i stanje u kojem se prikazuju u *canvas*-u, a pogled na scenu može se promijeniti pomoću jedne linije koda. Iscrtavanje primitiva na scenu izvodi se korištenjem vizualizatora koji obavlja WebGL pozive.

Budući da je orijentacija Cesium-a prvenstveno okrenuta prema geoprostornim sadržajima, nisu predviđene scene s više izvora svjetlosti pa Cesium koristi unaprijedni grafički protok za sjenčanje (engl. *forward shading pipeline*). Grafički pogon Cesiuma je jedinstven, jer koristi nekoliko piramida pogleda kako bi mogao podržavati pogled s velikih udaljenosti bez pojava nepravilnosti nastalih nepreciznim izračunima z-spremnika (engl. *z-fighting artifacts*).

Scene.render() se sastoji od 3 koraka:

- inicijalizacija: postavljanje stanja trenutnog okvira
- osvježavanje: sinkronizacija stanja primitiva sa stanjem vizualizatora, npr. spremnik vrhova i teksture
- vizualizacija: pozivi iscrtavanja za svaki primitiv

Kamera predstavlja pogled u virtualni svijet, odnosno stvara matricu pogleda koja transformira iz koordinata svijeta u koordinate oka. Kamera se automatski mijenja ovisno o ulaznim signalima miša preko komponente *ScreenSpaceCameraController*. Kamerom se može izravno upravljati i to najčešće kroz *CameraController*.

2.1.2.1.Dinamička scena

Dinamička scena je građena nad nižim slojevima te omogućuje dinamičku vizualizaciju podataka uglavnom baziranu na CZML jeziku. Ona omogućuje da se umjesto ručnog pozivanja osvježavanja (engl. *update*) primitiva u svakom okviru koristi jedan poziv osvježavanja. Podaci se spremaju u dinamičke objekte te se učitavanje i prikaz vrše iz vizualizatora koje pruža ovaj sloj.

Dolje prikazani isječak koda pokazuje kako učitati i vizualizirati CZML dokument u Cesium aplikaciju. Prvi korak uključuje stvaranje nekoliko objekata: scene (engl. *scene*), dinamičke kolekcije objekata (engl. *dynamicObjectCollection*), kolekcije Cesium vizualizatora (engl. *visualizers*) te sat (engl. *clock*) kako bi se upravljalo dinamičkom scenom. Zatim se učita i parsira CZML dokument, iz kojega se napuni dinamička kolekcija te izračunaju intervali animacija. Iza početnog postavljanja poziva se vrši osvježavanje trenutnog okvira na temelju trenutnog vremena.

```
var scene = new Scene(document.getElementById("canvas"));
var dynamicObjectCollection = new DynamicObjectCollection();
var visualizers = VisualizerCollection.createCzmlStandardCollection(scene,
            dynamicObjectCollection);
var clock = new Clock();
var czmlUrl = 'http://cesiumjs.org/someFile.czml';

getJson(czmlUrl).then(function(czml) {
            processCzml(czml, dynamicObjectCollection, czmlUrl);
            var availability = dynamicObjectCollection.
                                                computeAvailability();
            clock.startTime = availability.start;
            clock.stopTime = availability.stop;
});

var currentTime = clock.tick();
visualizers.update(currentTime);
```

2.1.3. Sloj primitiva

Prema definiciji primitiva u Cesium-u, primitiv je sve ono što ima definiranu funkciju osvježavanja i dodaje naredbe u listu naredbi scene. Odnose prema ostalim slojevima i položaj primitiva u grafičkom protočnom stogu prikazuje Slika 2-1.

2.1.3.1.Globus

Globus (engl. *Globe*) je primitiv koji prikazuje sferu: teren, slikovne slojeve i animiranu vodu. U grafičkom pogonu igre, ovaj primitiv bio odgovarao okruženju ili nivou. Velika razlika između igre i Cesium-a je ta da veličina okruženja i kretanje nisu ograničeni.

Cesium koristi četverostablo(engl. *quadtree*), u geografskim koordinatama, za ostvarenje hijerarhijske razine detalja (engl. *Hierarchiical level of detail*, HLOD). Primitiv četverostabla (engl. *QuadtreePrimitive*) još nije u potpunosti ostvaren te se očekuju značajna poboljšanja prilikom iscrtavanja 3D geometrije kada bude.

Tijekom rada aplikacije, ploče slikovnih slojeva se dinamički mapiraju na svaku ploču terena. Ovo omogućuje veću fleksibilnost, ali uz visoku cijenu kompleksnosti koda i dinamičke reprojekcije slikovnih slojeva za projekcije različite od osnovne. Po ovome se Cesium u ptpunosti razlikuje od grafičkog pogona igre koji učitava scenu koja je što je više moguće optimizirana, ali uz cijenu fleksibilnosti.

2.1.3.2.Model

Model predstavlja primitiv koji se odnosi na uobičajeni 3D model u računalnoj grafici. Cesium podržava jedino modele u glTF formatu, a omogućava pretvorbu Collada modela u glTF. Detaljnije o 3D modelima se nalazi u poglavlju 2.2.3.

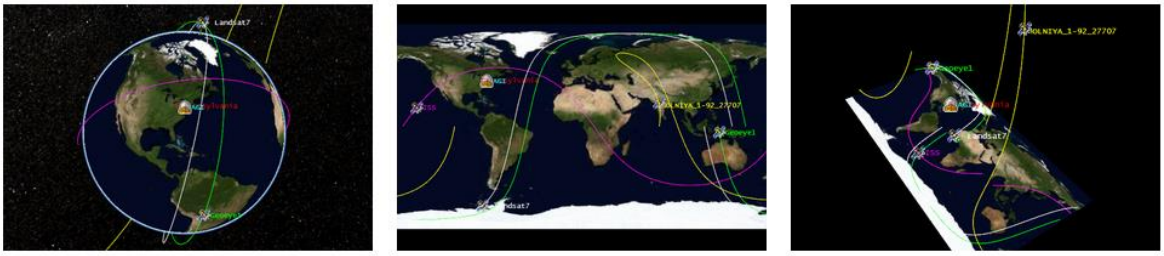
2.1.3.3. Ostali primitivi

U ostale primitive ubrajaju se velika skupina geometrija koje Cesium podržava, a iscrtava se na sceni. Također, Cesium ima visinski grafički protok (engl. *tesselation pipeline*), koji je prilagođen i iscrtavanju objekata na površini WGS84 elipsoida. Visinskim grafičkim pogonom se upravlja dinamički, a on omogućuje potporu geoprostorim standardima i formatima koji točno opisuju geometrijske parametre.

Ako su geometrije zapisane u istom formatu vrhova, Cesium različite geometrije stavlja u isti spremnik vrhova. Programi za sjenčanje vrhova i fragmenata uključuju vizualizaciju visoke preciznosti GPU *Relative to Eye*, potporu za 2D, 3D i Columbusov pogled te prikazivanje ili sakrivanje geometrije.

2.2 Mogućnosti Cesium-a

Cesium omogućuje tri načina pogleda podataka unutar same aplikacije: 3D sferu, 2D plohu i Columbus pogled (2.5D). Primjere za svaki od pogleda prikazuje Slika 2-2.



Slika 2-2 Lijevo: 3D sfera, sredina: 2D ploha, desno: Columbus view

Cesium podržava dinamičku geoprostornu vizualizaciju. Dinamička vizualizacija scena postiže se CZML jezikom, a mogu se prikazivati i tereni visoke rezolucije. Za prikaz podataka podržani su standardni formati: KML, GeoJSON i TopoJSON, a moguće je učitati prikaze slojeva iz različitih izvora, npr.: WMS, TMS, WMTS, *OpenStreetMap, Bing Maps, ArcGIS MapServer, Google Earth Enterprise*. Prikaz 3D modela s animacijama podržava COLLADA i glTF formate, a moguće je prikazati i razne geometrijske oblike. Također, za bolje efekte vizualizacije moguće je dodati atmosferu, sunce, mjesec, zvijezde i vodu.

Vrlo koristan alat je Cesium *Widget* preko kojeg se vrlo jednostavno mogu prikazati podaci. Kroz njega postoji mogućnost korištenja vremenske crte i animacija za upravljanje simulacijama kroz vrijeme, odabira baznog sloj na kojem će se tereni prikazivati, promjene načina pogleda, odabira pojedinih objekata te prikaza informacija o njima, upravljanj kamerom, te *Inspector Widget* za napredniji postupak ispitnog pokretanja (engl. d*ebugging*).

Za Cesium postoje i dodatne biblioteke koje proširuju njegove mogućnosti. Trenutno su dostupne sljedeće biblioteke:

- *Cesium Sensors* – priključak za vizualizaciju volumena

- *CesiumVR* – integracija Cesium-a za virtualnu stvarnost
- *GeoServerTerrainProvider* – omogućuje korištenje terena s *GeoServer*-a
- *DrawHelper* – alat koji pomaže crtanju jednostavnih oblika i poligona
- *Leap* – podrška za *Leap Motion*
- *Materials Pack* - skup materijala za korištenje pri prikazu
- *Assets* – skup slojeva, tekstura i grafičkih podataka prilagođen uporabi u Cesiumu

2.2.1. Prikaz prostornih podataka

Prikaz prostornih podataka može se ostvariti na dva načina, kroz *Primitive API* i *Entity API*. *Primitive API* zahtjeva minimalnu razinu apstrakcije te se očekuje da programer bude upoznat s pojmovima računalne grafike. Strukturiran je da omogući najbolje karakteristike i fleksibilnost za svaki tip vizualizacije. Nasuprot tome, *Entity API* koristi alate *Primitive API*-ija na višoj razini apstrakcije kroz jedinstvenu strukturu podataka - entitet (engl. Entity). *Entity API* se više fokusira na samu prezentaciju podataka, nego na mehanizme vizualizacije te nudi sučelje za fleksibilnu vizualizaciju.

Da bi se upogonila osnovna Cesium aplikacija (engl. *Cesium widget*) potrebno je upisati nekoliko linija koda. Najprije upisati put do Cesium.js skripte, te u CSS-u uključiti *Cesium Viewer widget*. Osigurati div element u kojem će se *Viewer* prikazivati i napraviti instancu *Viewer-a*.

```
<script src="Cesium/Cesium.js"></script>
//unutar HTML body
<div id="cesiumContainer"></div>
var viewer = new Cesium.CesiumViewer('cesiumContainer');
//u CSS skripti
@import url(Cesium/Widgets/widgets.css);
```

Viewer-u se entitet dodaje funkcijom *entities.add*(). Objekt koji se predaje zapravo je niz opcionalnih parametara koji su postavljeni na početne vrijednosti, na primjer:

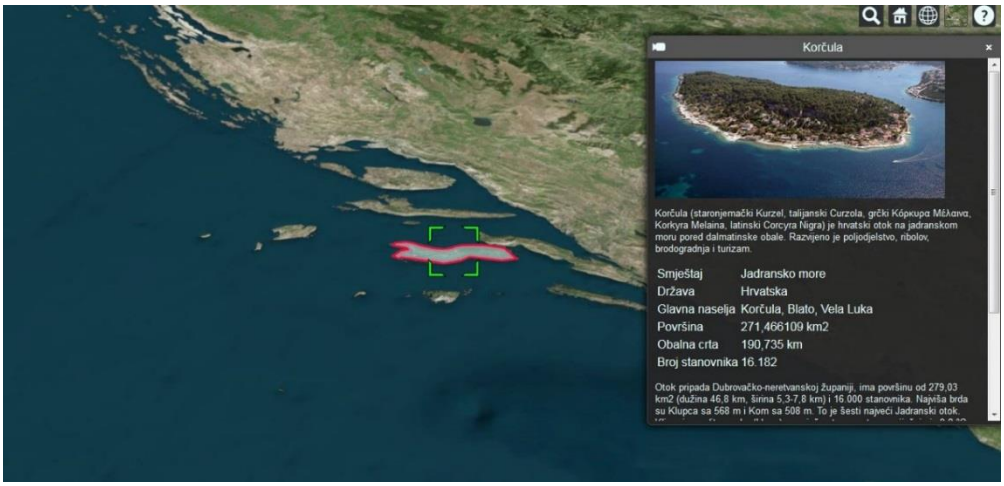
```
var korcula = viewer.entities.add({
                name : 'Korčula',
                polygon : {
                        hierarchy : Cesium.Cartesian3.fromDegreesArray([
                                16.820553, 42.959999, ..., 16.820553, 42.959999
                          ]),
                material : new Cesium.Color(0.6, 0.8, 0.8, 0.6),
                outline : false
                 }
});
```

Entitetu je pridruženo ime koje će se prikazati na ekranu, lista koordinata koje se iz geografske širine i dužine konvertiraju u 3D kartezijev sustav, materijal kojim će poligon biti prikazan te je za vanjski obrub poligona postavljeno da se ne prikazuje.

Osim poligona, kroz *EntityAPI* moguće je na sličan način dodati niz oblika i volumena, npr. elipsu, poliliniju, polilinijski volumen, pravokutnik, sferu, elipsoid, zidove itd. Materijal kojim se ispunjava poligon moguće je na više načina definirati. U gornjem primjeru definirana je CZML boja. Konstruktoru boje predaju se četiri parametra, redom crvena, zelena, plava i alfa parametar (prozirnost). Također, moguće je koristiti CZML predefinirane boje, čije se vrijednosti mogu vidjeti u dokumentaciji. U sljedećem primjeru dodan je obrub poligonu koristeći CZML boje.

```
var granice = viewer.entities.add({
                polyline:{
                        positions : ... ,
                        width : 5,
                        material : Cesium.Color.CRIMSON
                }
});
```

Entitetu je, također, moguće dodati opis koji se prikazuje kada se klikne na njega. Izgled opisa moguće je uređivati tako da se upiše kao dio HTML5 dokumenta. Primjer poligona definiranog prikazanim kodom uz dodatak opisa prikazuje Slika 2-3.



Slika 2-3 Prikaz poligona na karti s njegovim opisom

Slika 2-4 Prikaz labela

Također, vrlo je jednostavno dodati oznake te labele nazivom objekta. Entitetu se dodaju svojstva kako će točka i labela izgledati. Primjer definiranja oznake i natpisa je u sljedećem dijelu koda, a izgled prikazuje Slika 2-4.

```
var gradVelaLuka = viewer.entities.add({
            name : 'Vela Luka',
            position : Cesium.Cartesian3.fromDegrees(16.72, 42.96),
            point : {
                  pixelSize : 5, color : Cesium.Color.MEDIUMBLUE,
                  outlineColor : Cesium.Color.WHITE, outlineWidth : 2
            },
            label : {
                  text : 'Vela Luka', font : '12pt calibri',
                  style: Cesium.LabelStyle.FILL_AND_OUTLINE,
                  outlineWidth : 2,
                  verticalOrigin : Cesium.VerticalOrigin.BOTTOM,
                  pixelOffset : new Cesium.Cartesian2(0, -10)
            }
           });
```

2.2.2. Prikaz slojeva slikovnih podataka

Cesium podržava prikaz mapa i slojeva slikovnih podataka visoke rezolucije koje se mogu dohvatiti iz nekoliko standardnih servisa. Slojevi mogu biti poredani i spojeni, a svakom sloju mogu se dinamički mijenjati svojstva.

Servisi koji su podržani u Cesiumu su sljedeći: WMS (*Web Map Service*), TMS (*Tile Map Service*), WMTS (*OpenGIS Web Map Tile Service*), *OpenStreetMap*, *BingMaps*, *Esri ArcGIS MapServer, Google Earth Enterprise, Standard image files, Tile coordinates*.

Pojedinim servisima se pristupa kroz *ImageryProvider* sučelje. Većina sučelja ima konstruktor kojem je moguće predati sljedeće parametre:

- *url* – jedini obavezni parametar, označava url do slike
- *extent* – pravokutnik koji označava dužinu-širinu koju slika treba prekrivati
- *credit* – string koji označava adresu za prikaz loga servisa
- *proxy* – proxy koji se koristi kod zahtjeva koji se upućuju servisu

U sljedećem primjeru koriste se slojevi s dva servisa. Prilikom konstruiranja *Viewer widgeta* moguće je postaviti drugačiji osnovni sloj.

```
var viewer = new Cesium.Viewer('cesiumContainer', {
            imageryProvider : new Cesium.ArcGisMapServerImageryProvider({
                    url :
                    'http://server.arcgisonline.com/arcgis/
                          rest/services/NatGeo_World_Map/MapServer'
            }),
            baseLayerPicker : false
});
```

Ovim kodom je za osnovni sloj odabran sloj slikovnih podataka koji prikazuje Slika 2-5.

Slika 2-5 Prikaz sloja s ArcGIS MapServer-a

Također, moguće je u prikaz dodati još slojeva. Slojevi dodani kasnije prekrivaju slojeve koji su ranije dodani. Poredak slojeva se može mijenjati, ali se oni mogu i stapati. Sljedeći primjer pokazuje kako se slojevi mogu kombinirati te je gornjem sloju promijenjena prozirnost te svjetlina. Zajednički prikaz slojeva zadan isječkom koda daje Slika 2-6.

```
var layers = viewer.scene.imageryLayers;
var blackMarble = layers.addImageryProvider(
                    new Cesium.TileMapServiceImageryProvider({
                     url : '//cesiumjs.org/tilesets/imagery/blackmarble',
                     maximumLevel : 8,
                    credit : 'Black Marble imagery courtesy NASA Earth
                                        Observatory'
}));
blackMarble.alpha = 0.4;
blackMarble.brightness = 3.0;
```

Slika 2-6 Prikaz više slojeva iz različitih servisa

2.2.3. 3D modeli

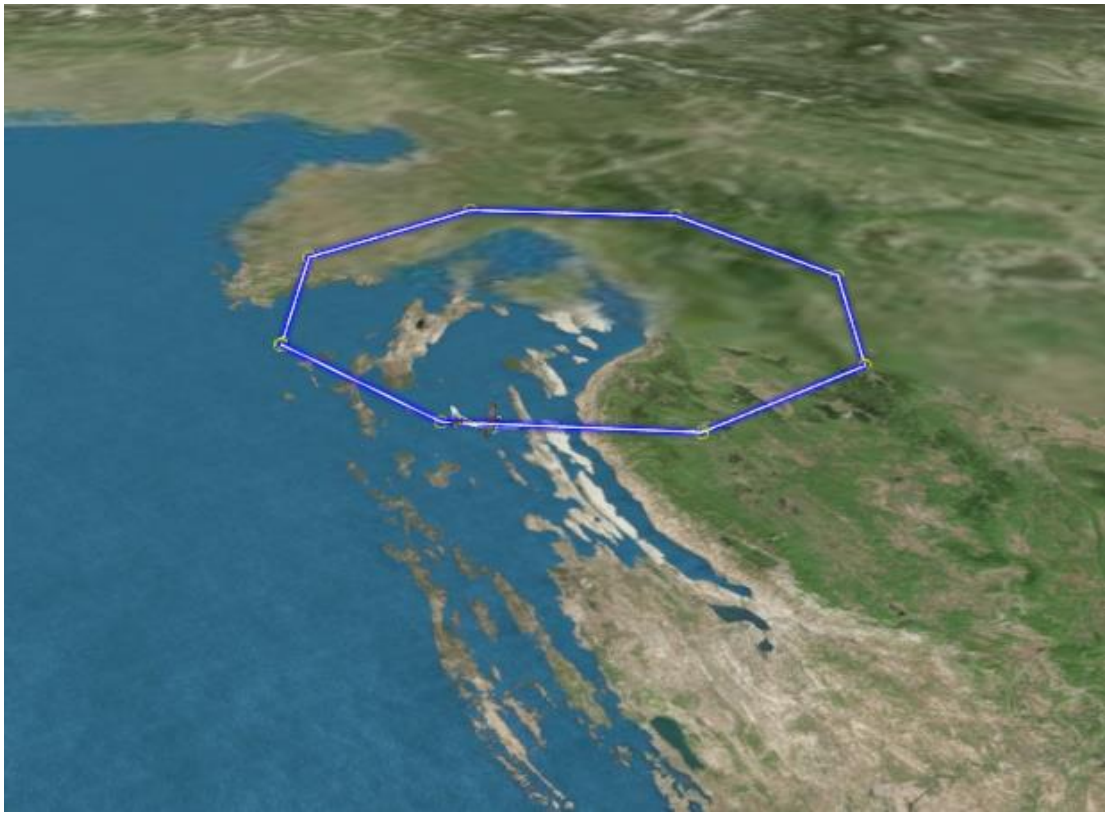
Cesium podržava 3D modele, zajedno s njihovim animacijama po ključnim okvirima, dodavanje kože te odabirom pojedinih dijelova koristeći glTF format. glTF format je industrijski standardizirani format za 3D modele na webu koji se sve više koristi. Cesium putem web sučelja nudi alat za konverziju COLLADA modela u glTF format. Model se u scenu dodaje kao primitiv kroz funkciju *Cesium.Model.fromGltf*, a modelu se predaje matrica koja određuje njegov koordinatni sustav. Ako model ima definirane animacije potrebno ih je aktivirati funkcijom *addAll*, a animacije se pokreću funkcijom *Cesium.modelAnimationLoop.REPEAT* te se prikazuju sve dok postoje u kolekciji *activeAnimations*. Učitani model aviona prikazuje Slika 2-7.

```
var model = scene.primitives.add(Cesium.Model.fromGltf({
            url: '../../SampleData/models/CesiumAir/CesiumAir.flTF',
            modelMatrix: modelMatrix,
            scale : 300.
}));
Cesium.when(mode.readyPromise).then(function(model) {
    model.activeAnimations.addAll({
        loop : Cesium.ModelAnimationLoop.REPEAT
    });
});
```

Slika 2-7 Model aviona učitan iz glTF formata

Animacije se osim učitavanjem iz modela mogu zadati kroz kod. Koriste se početni i krajnji datum Julijanskog kalendara te se oni postave kao vrijeme početka i vrijeme zaustavljanja animacije, a vremenska crta se namjesti da prikazuje zadane datume. Zatim se zada lista pozicija kojima se element treba kretati. Kada se entitet dodaje *Viewer widget-u* potrebno je upisati vrijeme kada je dostupan, poziciju, orijentaciju te učitati model. Da bi animacija bila glatka moguće je koristiti interpolacijski algoritam kao u sljedećem primjeru, a točke između kojih se interpolira i kroz koje avion prolazi prikazuje Slika 2-8.

```
entity.position.setInterpolationOptions({
interpolationDegree : 15,
interpolationAlgorithm : Cesium.LinearApproximation
});
```



Slika 2-8 Animacija aviona, te točke oko kojih se interpolira

2.2.4. Kamera

Kamera kontrolira pogled na scenu. Postoji mnogo načina za manipuliranje kamerom poput rotacije, zumiranja i leta do željene pozicije. Postoje ugrađeni funkcije za registriranje događaja pri interakciji s kamerom i API kojim se programski može manipulirati kamerom.

```
camera.setView({
    positionCartografic : new Cesium.Cartesian3(x, y, z),
    position : Cesium.Cartesian3.fromDegrees(longitude, latitude, height),
    heading : headingAngle,
    pitch : pitchAngle,
    roll : rollAngle
});
```

Ugrađeni događaji i rezultati akcija su sljedeći:

- lijevi klik miša i povlačenje – rotira kameru oko sfere u 3D pogledu, a u 2D i 2.5D pogledu translatira kameru
- desni klik miša i povlačenje – približava i udaljava kameru od gledišta
- okretanje kotača na mišu – isto približava i udaljava kameru od gledišta
- srednji klik i povlačenje – rotira kameru oko točke na površini sfere

Funkcijom *setView* se programski određuje pozicija i orijentacija kamere. Argumenti koje funkcija prima su pozicija, dubina, smjer i nagib. Pozicija može biti zadana u kartezijevom 3D sustavu ili kartografskom sustavu, dok ostala 3 argumenta trebaju biti zadana u radijanima. Smjer je zadan kao rotacija od početnog usmjerenja prema sjeveru, a pozitivni smjer rotacije je prema istoku. Pozitivan nagib označavaju kutovi iznad ravnine, a negativni ispod ravnine. Dubina označava prvu rotaciju koja se primjenjuje u odnosu na lokalnu istočnu os.

Primjer definiranja pogleda kamere daje sljedeći isječak koda. Ako je pozicija kamere zadana na oba moguća načina preferirati će se ona zadana u kartezijevom sustavu, a svi parametri koji nisu zadani kroz argumente bit će namješteni na trenutnu poziciju kamere.

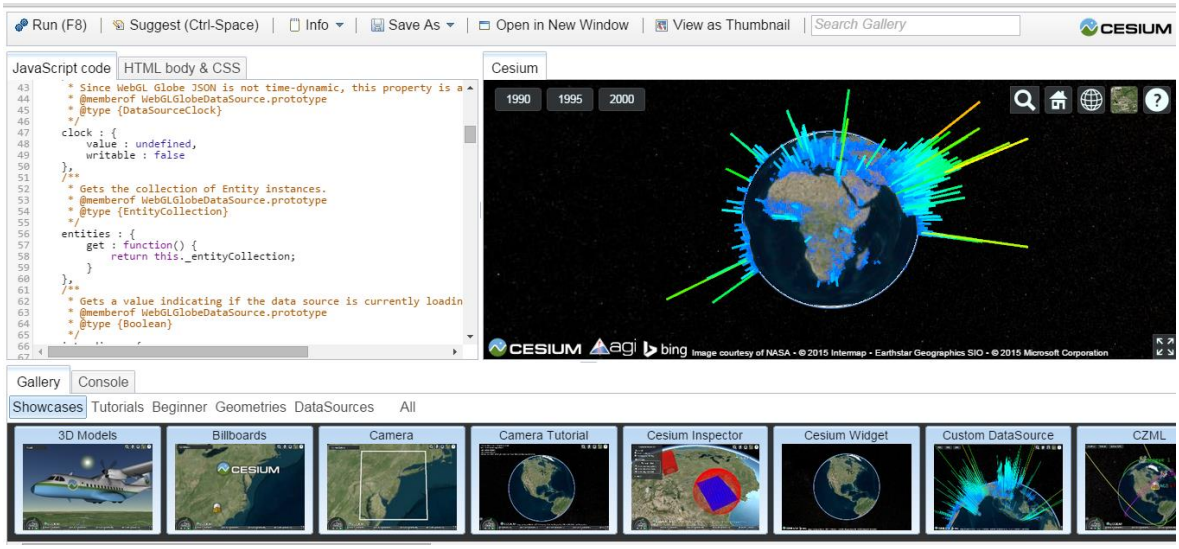
2.3 Prednosti Cesium-a

Cesium je Javascript biblioteka otvorenog koda pod Apache 2.0 dozvolom. Namijenjen je prikazu geoinformacijskih podataka unutar web preglednika bez potrebe za instalacijom dodataka. Korištenje ove biblioteke je u potpunosti besplatno za komercijalnu i nekomercijalnu upotrebu, a svatko može dodavati nove funkcionalnosti i uređivati postojeće.

Postoji opširna i obuhvatna dokumentacija koda s primjerima korištenja funkcija, te pokazni primjeri programskog koda za implementaciju određenih funkcionalnosti. Također, postoji forum s diskusijama na kojem zajednica pomaže sa svojim savjetima.

Podržan je uvoz podataka iz različitih vanjskih izvora, te se potrebni matematički izračuni (koji su ponekad prilično složeni) dosta brzo izvrše. Kontrola i kretanja kamere su jako dobro izvedeni te nije teško koristiti postojeće funkcije te implementirati vlastite za kontroliranje kamere.

S obzirom na to da je Cesium izgrađen direktno nad WebGL-om olakšano je učitavanje većih kompleksnih geometrija. Dodatno, uz pomoć WebGL-a mogući su dodatni zahtjevniji izračuni na sferi.



Slika 2-9 Izgled Sandcastle-a za kodiranje „uživo“

Razvoj osnovnih aplikacija za prikaz podataka u Cesium-u je dosta lagan zahvaljujući dobro strukturiranom kodu i dosta detaljnom dokumentacijom. Također postoji mogućnost da se koristi *Sandcastle* koji omogućuje kodiranje „uživo“ kroz preglednik. Unutar preglednika se otvara sučelje koje se sastoji od dijela u koji se upisuje kod s lijeve strane, a s desne strane se vidi rezultat koji se dobije tim kodom. Primjer kako izgleda kodiranje sa *Sandcastle*-om prikazuje Slika 2-9.

Cesium omogućava učitavanje geoinformacijskih podataka iz različitih izvora i u formatima koji su većinom standardni za prikaz prostornih podataka, te manipulaciju njima. To ga čini prikladnim za korištenje u aplikacijama kojima je prvenstveni cilj prikazati izgled nekog područja (reljefa) i pritom prikazati neke dodatne informacije o tom području. Njegova prednost je što nema pretjerane potrebe za skidanjem i povezivanjem dodatnih biblioteka kako bi različiti skupovi podataka međusobno dobro funkcionirali.

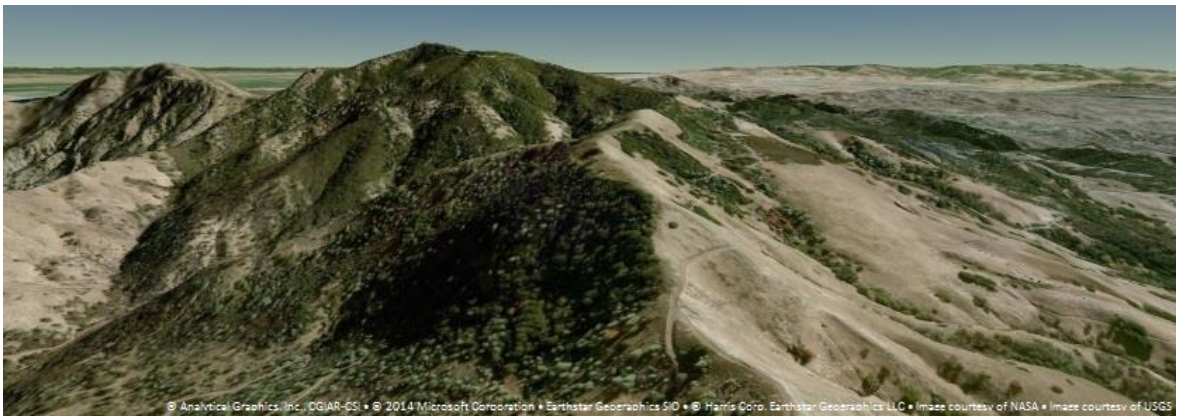
U konačnici, postavljanje aplikacije u pogon na vlastitom poslužitelju je izrazito jednostavno. Ono što je nužno i jedino potrebno je uključiti odgovarajuće Javascript i css datoteke.

3. OSTVARENJE TERENA I SLOJEVA

3.1. Teren

Teren opisuje oblik površinske geometrije objekta koji služi kao podloga. Skup podataka koji predstavlja teren poznat je kao digitalni visinski model, odnosno DEM (*Digital Elevation Model*). DEM je 3D reprezentacija površine terena koja nastaje iz visinskih podataka o samom terenu. Geometrija terena najčešće je opisana visinskom mapom i mrežom trokuta, a sam površinski izgled može biti osjenčan ili opisan nekim slikovnim slojem.

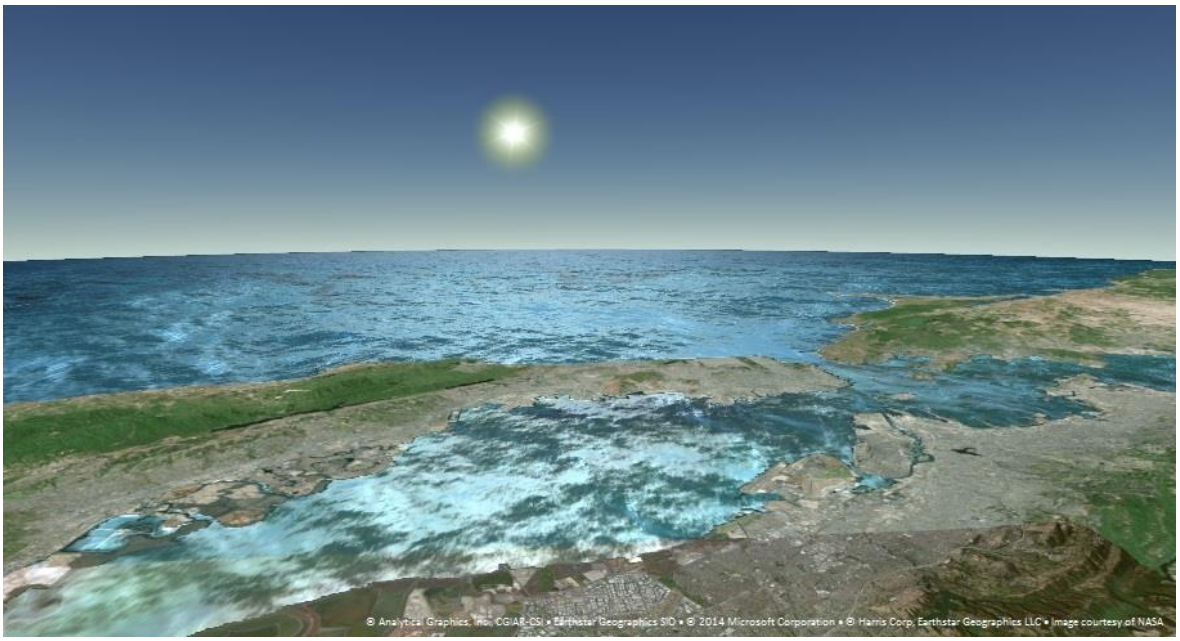
Cesium omogućuje povezivanje s poslužiteljima koji pružaju podatke o geometriji terena. Također, omogućuje korištenje dvije vrste ugrađenih terena: *STK World Terrain* i *Small Terrain. STK World Terrain* je teren visoke rezolucije koji se sastoji od skupa ploča s visinskim podacima o površini cijele Zemlje, a izgled dijela terena prikazuje Slika 3-1. Podaci su opisani mrežom trokuta, a zapisani u formatu *quantized-mesh-1.0*.



Slika 3-1 STK World Terrain

Ako se STK World Terrain želi koristiti kroz Cesium potrebno je definirati *TerrainProvider* i kao url postaviti *'//assets.agi.com/stk-terrain/world*'.

Small Terrain je nešto niže rezolucije, a češće se koristio u početcima razvoja Cesium-a. Podaci su zapisani kao visinske mape, u formatu *heightmap-1.0*. Prilikom prikaza *Small terrain*-a izgled je lošiji, odnosno "grublji" od *STK World Terrain*-a, a kao primjer može se vidjeti Slika 3-2.



Slika 3-2 Small Terrain

Postavljanje ovog terena kao podloge ne razlikuje se od onog kako se koristi *STK-World Terrain*, a primjer korištenja s *Cesium Widgetom* može se vidjeti u sljedećem dijelu koda:

```
var widget = new Cesium.CesiumWidget('cesiumContainer', {
            terrainProvider :    new Cesium.CesiumTerrainProvider({
                    url:'//cesiumjs.org/tilesets/terrain/smallterrain'
                    })
});
```

Podaci za *STK-World Terrain* se puno češće osvježavaju, a moguće je korištenje nekih dodatnih funkcionalnosti, poput osvjetljena na terenu i uključivanja vodene maske. Opis geomerije *STK-World Terrain-a* dan je mrežom trokuta pa stoga može biti bolje iskorišten pri dodavanju više slojeva. U novijim verzijama Cesium-a *STK-World Terrain* će nadvladati *Small Terrain*.

3.1.1. Format heightmap-1.0 za zapis geometrije terena

Skup ploča koje predstavljaju teren u ovom formatu zapisan je kao četvero stablo visinskih mapa. Svaka ploča u svom nazivu ima ekstenziju *terrain*. Ako je, na primjer, poveznica do mape u kojoj se nalaze ploče '*//tileset/*', onda se adrese datoteka u dva korijenska direktorija nalaze na sljedećim poveznicama:

- (-180 deg, -90 deg) - (0 deg, 90 deg) - *'//tileset/0/0/0.terrain'*
- (0 deg, -90 deg) - (180 deg, 90 deg) - *'//tileset/0/1/0.terrain'*

8 ploča koje pripadaju sljedećem nivo-u zapisa nalaze se na sljedećim poveznicama:

- (-180 deg, -90 deg) - (-90 deg, 0 deg) - *'//tileset/1/0/0.terrain'*
- (-90 deg, -90 deg) - (90 deg, 0 deg) - *'//tileset/1/1/0.terrain'*
- (0 deg, -90 deg) - (90 deg, 0 deg) - *'//tileset/1/2/0.terrain'*
- (90 deg, -90 deg) - (180 deg, 0 deg) - *'//tileset/1/3/0.terrain'*
- (-180 deg, 0 deg) - (0 deg, 90 deg) - *'//tileset/1/0/1.terrain'*
- (-90 deg, 0 deg) - (0 deg, 90 deg) - *'//tileset/1/1/1.terrain'*
- (0 deg, 0 deg) - (90 deg, 90 deg) - *'//tileset/1/2/1.terrain'*
- (90 deg, 0 deg) - (180 deg, 90 deg) - *'//tileset/1/3/1.terrain'*

Svaka ploča sadrži podatke o 65x65 vrhova i preklapa se sa susjednim pločama na rubovima, odnosno kod susjednih ploča podaci o visinama na rubovima koji su jedan kraj drugog su jednaki.

Podaci sa servera dolaze u sljedećem obliku:

- prvi dio, ujedno i najvažniji, sastoji se od jednostavnog niza 16 bitnih cijelih brojeva, u *little-endian*-u. Svaki zapis u nizu predstavlja visinu u jednom vrhu, a zapisi su poredani od sjevera prema jugu i od zapada prema istoku. Prvi zapis započinje sa sjeverozapadnim uglom, a nastavlja se prema istočnim lokacijama. Svaka jedinica visine predstavlja jedinu petinu metra iznad -1000 m. Ukupna veličina poslanih podataka iznosi 65*65*2 = 8450 okteta.

- Nakon visinskih podataka nalazi se oktet koji predstavlja koje child tiles su prisutne na poslužitelju. Vrijednosti bitova su sljedeće:
    - jugozapad - bit 0 - vrijednost 1
    - jugoistok - bit 1 - vrijednost 2
    - sjeverozapad - bit 2 - vrijednost 4
    - sjeveroistok - bit 3 - vrijednost 8

  Ako je vrijednost bita postavljena na 1, očekuje se da će se na serveru nalaziti odgovarajući *.terrain* file, a ako ga nema, vratit će se kod greške *404*.

- Nakon maske child bit-ova, slijedi vodena maska. Vodena maska će biti 1 oktet ukoliko ploča čini samo kopno ili samo vodu, ili će biti sastavljena od 256*256*1 = 65536 okteta ako je sastavljena djelomično od kopna, a djelomično od vode. Vrijednost bit maske će biti 0 ako predstavlja kopno, 255 ako predstavlja vodu, a dopuštene su i vrijednosti između 0 i 255 radi *anti-aliasing*-a obalne linije (ali za sada još nije prisutna u podacima).

3.1.2. Format Quantized-mesh-1.0 za zapis geometrije terena

Zapis ploča i vrhova odgovara četvero stablu koji je opisan za prvi format. Ono u čemu se ova dva formata razlikuju je sam zapis podataka po pločama, odnosno zapis vrhova. Bitno je napomenuti da je prilikom slanja zahtjeva na poslužitelj nužno uključiti sljedeće HTTP zaglavlje bi osigurali da poslužitelj vrati željenu reprezentaciju zapisa ploča:

Accept: application/vnd.quantized-mesh,application/octet-stream;q=0.9

Svaka ploča je posebno kodirana mreža trokuta, a zapisi rubnih vrhova preklapaju se sa susjednim pločama, odnosno najistočniji vrhovi zapadne ploče imaju istu geografsku dužinu kao i najzapadniji vrhovi istočne ploče.

Zapis ploča je u *little endian*-u, a sastoji se od nekoliko dijelova. Zapisi *double* i *float* brojeva su u IEEE 754 zapisu realnih brojeva s decimalnom točkom, *double* 64 bitni, a *float* 32-bitni. Struktura zaglavlja (engl. *header*) zapisa ploče je sljedeća:

```
struct QuantizedMeshHeader {
        double CenterX;
        double CenterY;
        double CenterZ;
        //centar ploče izražen u koordinatama u odnosu na centar Zemlje
        float MinimumHeigt;
        float MaximumHeight;
        // minimalna i maksimalna visina u području koje ova ploča
        prekriva
        double BoundingSphereCenterX;
        double BoundingSphereCenterY;
        double BoundingSphereCenterZ;//centar sfere opisane ploči
        double BoundingSphereRadius;//promjer sfere u metrima
        double HorizontOcclusionPointX;
        double HorizontOcclusionPointY;
        double HorizontOcclusionPointY; //točka zaklanjanja horizonta
}
```

Odmah nakon zaglavlja slijede podaci o vrhovima. *Unsigned int* format je 32 bitni cijeli broj bez predznaka, a *unsigned short* je 16 bitni cijeli broj bez predznaka. Struktura podataka koja predstavlja jedan vrh dana je sljedećim isječkom:

```
struct vertexData {
        unsigned int vertexCount;
        unsigned short u[vertexCount];
        unsigned short v[vertexCount];
        unsigned short z[vertexCount];
}
```

Varijabla *vertexCount* predstavlja broj članova u nizovima koji slijede. Tri niza su zig-zag kodirana tako da imaju vrijednosti *small integer*-a, bezobzira na predznak. Dekodiranje se stoga obavlja prema sljedećoj formuli:

decoded = (encodedValue >> 1) ^ (-(encodedValue&1))

Nakon što su podaci iz nizova kodirani imaju sljedeće značenje:

- *u* - horizontalna koordinata vrha na ploči. Kada ima vrijednost 0 znači da se nalazi na zapadnom rubu ploče, a vrijednost 32767 označava da se nalazi na istočnom rubu. Ostale vrijednosti dužine za vrh dobivaju se linearnom interpolacijom između istočnog i zapadnog ruba ploče.
- *v* - vertikalna koordinata vrha na ploči. Kada ima vrijednost 0 znači da se nalazi na južnom rubu ploče, a vrijednost 32767 označava da se nalazi na sjevernom rubu. Ostale vrijednosti dužine za vrh dobivaju se linearnom interpolacijom između južnog i sjevernog ruba ploče.
- *height* - visina vrha na ploči. Kada je jednaka 0 označava da je visina jedna minimalnoj visini ploče koja je specificirana u zaglavlju, a ako je 32767 onda je jednaka maksimalnoj visini ploče. Ostale vrijednosti visine dobivaju se linearnom interpolacijom između maksimalne i minimalne visine.

Nakon podataka o koordinatama vrhovima slijede podaci o indeksima. Indeksi određuju kako su podaci o vrhovima međusobno povezani u trokute. Ako ploča ima više od 65536 vrhova, onda se koristi *IndexData32* struktura kako bi se indeksi kodirali, a inače se koristi *IndexData16* struktura. Kako bi se osigurao ispravan poredak okteta *IndexData* struktura

ima odgovarajući *padding* od 2 okteta za 16 bitne, odnosno 4 okteta za 32bitne podatke. Izgled spomenutih struktura se može vidjeti u sljedećem isječku:

```
struct IndexData16 {
        unsigned int triangleCount;
        unsigned short indices[triangleCount*3];
}
struct IndexData32 {
        unsigned int triangleCount;
        unsigned short indices[triangleCount*3];
}
```

Indeksi su kodirani koristeći vodeni žig iz *webgl-loader*-a, a dekodiraju se na sljedeći način prikazan sljedećim isječkom koda:

```
var highest = 0;
for (var i = 0; i < indices.length; ++i) {
        var code = indices[i];
        indices[i] = highest - code;
        if (code === 0){
                ++highest;
        }
}
```

Svaka trojka indeksa određuje jedan trokut koji će se prikazati, a redoslijed zapisa je obrnut od smjera kazaljke na satu. Nakon liste indeksa trokuta slijede još četiri liste indeksa:

```
struct EdgeIndices16{
    unsigned int westVertexCount; unsigned short westIndices[westVertexCount];
    unsigned int southVertexCount;unsigned short southIndices[westVertexCount];
    unsigned int eastVertexCount; unsigned short eastIndices[westVertexCount];
    unsigned int northVertexCount unsigned short northIndices[westVertexCount];
}

struct EdgeIndices32{
    unsigned int westVertexCount; unsigned int westIndices[westVertexCount];
    unsigned int southVertexCount; unsigned int southIndices[westVertexCount];
    unsigned int eastVertexCount; unsigned int eastIndices[westVertexCount];
    unsigned int northVertexCount; unsigned int northIndices[westVertexCount];
}
```

Ove liste indeksa enumeriraju vrhove koji se nalaze na rubovima ploča. Liste su ovako zapisane jer je korisno znati koji su vrhovi na rubovima kako bi se mogle sakriti pukotine prilikom prilagode LOD-a.

Nakon podataka o indeksima mogu slijediti opcionalni dodatni podaci. Svaki dodatak započinje s *ExtensionHeader*-om, a sastoji se od jedinstvenog identifikatora i veličine ekstenzije u bajtovima. Unsigned char predstavlja 8-bitni cijeli broj bez predznaka.

```
struct ExtensionHeader {
        unsigned char extensionId;
        unsgned int extensionLength;
}
```

Svaki novi dodatak ima svoj jedinstveni identifikator, a ako nema definiranih dodataka, onda se *ExtensionHeader* izostavlja. Dodatci mogu biti zahtijevani s klijentske strane, a zapis dodataka se odvaja s '-', pa na primjer, ako klijent traži normale vrhova i vodenu masku koristit će sljedeće zaglavlje u zahtjevu:

Accept:'application/vnd.quantized-mesh;extensions=octvertexnormals-watermask'

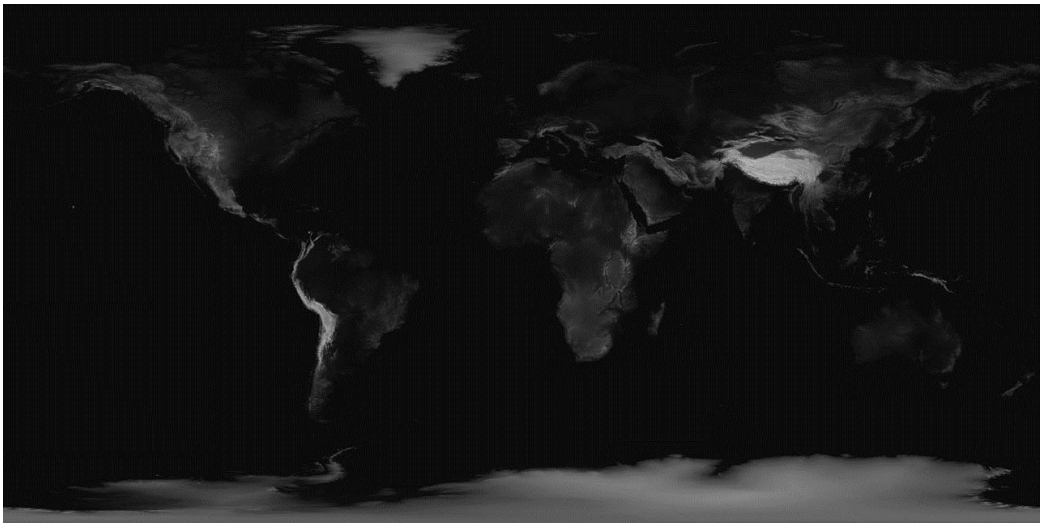
Za format *quantized-mesh* mogu biti definirane sljedeće dodatci:

- Terensko osvjetljenje
    - ime : *Oct-Encoded Per-Vertex Normals*
    - id: 1
    - dodaje osvjetljenje u vrhu. Svaka normala vrha koristi *oct-encoding* za kompresiju tradicionalnog 96-bitnog *float* zapisa x,y,z u 16-bitni x,y zapis.
    - HTTP zaglavlje:
      Accept:'application/vnd.quantized-mesh;extensions=octvertexnormals'
- Vodena maska
    - ime: Water mask
    - id: 2
    - Zapis podataka o obalnoj liniji za prikaz vodenih efekata. Vodena maska je veličine 1 oktet ako je ploča u potpunosti sastavljena od kopna ili vode, ili je veličine 256*256*1 = 65536 okteta ako je dio kopno, a dio voda. Ako je vrijednost u maski jednaka 0 onda predstavlja kopno, a ako je 255 onda predstavlja vodu. Vrijednosti vodene maske su definirane od sjevera prema jugu i od zapada prema istoku, a prva vrijednost predstavlja vrijednost u sjeverozapadnom uglu ploče. Vrijednosti između 0 i 255 također su dopuštene u svrhu *anti-aliasinga*.

3.1.3 Dobivanje skupa ploča s podacima

Da bi se dobili podaci za stvaranje 3D terena potrebno je imati digitalni visinski model (engl. *Digital Elevation Model*) koji predstavlja 3D reprezentaciju terena, a stvoren je na temelju visinskih podataka o području koje predstavlja. Primjer izgleda DEM-a prikazuje Slika 3-3.

DEM podaci se najčešće mogu pronaći na geodetskim web stranicama na razinama države koje vrlo često sadrže i mnoštvo drugih informacija o reljefima i područjima koja opisuju, ali se razlikuju u kvaliteti. Kvaliteta DEM-a se određuje prema točnosti visinskog podatka u svakom slikovnom elementu te koliko je točna morfologija koju predstavlja. Nekoliko važnih čimbenika koji utječu na kvalitetu su: grubost terena, gustoća prikupljanja podataka, veličina slikovnog elementa, odnosno rešetke i interpolacijski algoritam.



Slika 3-3 Izgled DEM-a za svijet - grayscale - 8-bit

Osim postojećih DEM datoteka moguće je iz visinskih podataka napraviti vlastiti. Postupak je sljedeći:

- u jednu datoteku popisati sve podatake u zapisu 3D točke (x, y, z)
- podaci iz datoteke učitati u *GlobalMapper* koji je dostupan i opisan na sljedećoj poveznici *http://www.bluemarblegeo.com/products/global-mapper.php*
- podatke iz *GlobalMapper-a* izvesti u željenom obliku, za Cesium je potrebno odabrati stvaranje visinske rešetke (engl. *Create Elevation Grid*) u sivim tonovima (engl. *grayscale*) i izvesti kao 32-bitnu Geotiff datoteku, odnosno DEM datoteku

Iz Geotiff datoteke potrebno je generirati skup ploča u već opisanim formatima kako bi s poslužitelja Cesium-u poslali podatke koje on podržava. Za generiranje podataka koji su prikladniji za prikaz velikih terena, *quantized-mesh-1.0*, ne postoji besplatan alat, ali može se koristiti *STK Terrain Server* koji je dostupan na sljedećoj poveznici:

http://www.agi.com/products/stk/terrain-server

Generiranje skupa ploča u formatu *heightmap-1.0* je moguće preko alata koji se poziva iz naredbenog retka, a dostupan je na sljedećoj poveznici:

https://github.com/geo-data/cesium-terrain-builder

Upute za instalaciju i pokretanje dostupne su također na navedenoj poveznici. Preporuka je instalirati alat na neko *Unix* okruženje, uz važnu napomenu da u okruženju mora biti instaliran GDAL-2.0.0. Primjer kako iz DEM podataka dobiti skup terena pokazuje sljedeći poziv:

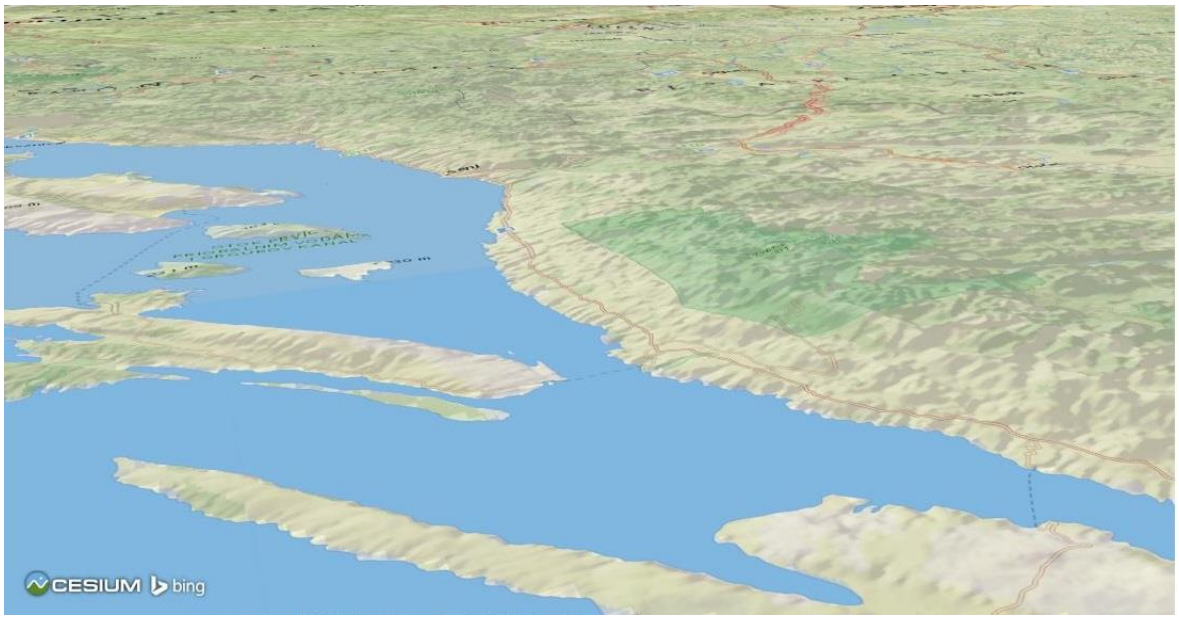
```
ctb-tile --output-dir ./teren_ploce dem.tif
```

Ovom naredbom se stvori direktorij */teren_ploce* u kojem se nalazi mnoštvo poddirektorija s podacima oblika *redni_broj.terrain* u formatu *heightmap-1.0*, a opis strukture direktorija i poddirektorija je opisan u poglavlju 3.1.1. Korijenski direktoriji i datoteke u njima moraju nužno postojati. S obzirom na to da *ctb-tile* ne stvara praznu nultu ploču, nju je potrebno u direktorij 0 i nazvati ju 0.terrain. Također, u direktoij */teren_ploce* potrebno je dodati *layer.json* datoteku sljedećeg sadržaja:

```
{
	"tilejson" : "2.1.0",
	"format" : "heightmap-1.0,
	"version" : "1.0.0",
	"scheme" : "tms",
	"tiles" : ["{z}/{x}/{y}.terrain?v={version}"]
}
```

Da bi se teren prikazao potrebno je *CesiumWidget*-u ili *Viewer*-u postaviti *terrainProvider*. On se može postaviti prilikom inicijalizacije Cesium aplikacije ili kasnije u kodu. Inicijalizacija Cesium aplikacije koja koristi teren, uključuje i odabir sloja slikovnih podataka kao bazne podloge za prikaz terena. Primjer inicijalizacije koja koristi teren iz direktorija */teren_ploce* je dan sljedećim isječkom koda, a izgled s pripadajućim slojem slikovnih podataka prikazuje Slika 3-4.

```
var viewer = new Cesium.Viewer('cesiumContainer', {
      imageryProvider : new Cesium.ArcGisMapServerImagerProvider({
            url : 'http://server.arcgisonline.com/arcgis/rest/services
                  /NatGeo_World_Map/MapServer',
      baseLayerPicker : false,
      terrainProvider : new Cesium.CesiumTerrainProvider({
      url : '/teren_ploce'
      })
});
```



Slika 3-4 Prikaz terena i GEO baznog sloja

3.2. Ostvarenje i prikaz terena i slojeva

3.2.1. WMS zahtjevi

WMS (*engl*. *Web Map Service*) je standarni protokol za posluživanje georeferenciranih karata od strane poslužitelja. Georeferencirana karta na web-u sastoji se od slika dijelova karte, koje nazivamo skupom ploča. Klijent šalje zahtjev poslužitelju za skupom ploča, a server stvara slike ovisno o parametrima koje je dobio od klijenta kroz zahtjev te ih šalje klijentu. Izvor od kojeg server stvara slike ne mora nužno biti slika, već mogu biti vektorski podaci, rasterski podaci ili njihova kombinacija.

Primjer zahtjeva koji se šalje na poslužitelj je sljedeći:

'http://localhost/geoserver/ws_svijet/wms?service=WMS&version=1.1.1&request=GetMap&styles=point&layers=airport&srs=EPSG:4326&bbox=-180,-90,1800,90&width=256&height=256'.

Adresu poslužitelja predstavlja *'http://localhost/geoserver/ws_svijet/wms'*, u ovom slučaju je riječ o geoserveru koji je podignut lokalno. Ostali parametri imaju sljedeće značenje:

- *service* – protokol koji se koristi
- *version* – verzija protokola koji se koristi
- *request* – zahtjev, *GetMap* označava da se želi dohvatiti slika karte
- *styles* – stil koji se koristi za uređivanje karte, odnosno za prikaz podataka iz izvora
- *layers* – naziv sloja kojeg se dohvaća s poslužitelja
- *srs* – projekcija u kojoj želimo da se karta iscrta
- *bbox* – koordinate unutar kojih želimo dobiti kartu
- *width, height* – svojstva slike

Učitavanje i dodavanje tako dohvaćenih slojeva na kartu u Cesium-u se obavlja kroz *ImageryProvider*-e koji su opisani u poglavlju 2.2.1.

3.2.2. Ostali izvori podataka

Razni formati u kojima često znaju biti geoinformacijski podaci u Cesium se mogu dodati kroz sučelje za izvor podataka (engl. *Data Source*). Na taj način je podražan uvoz podataka u sljedećim formatima: KML, GeoJson, TopoJson i CZML.

Sljedeći primjer pokazuje kako se učitavaju podaci iz KML formata, a dodatno se vrši i animacija prilikom iscrtavanja podataka. Podaci se učitavaju s adrese dane konstruktoru KML izvora podataka. Dohvaća se entitet u kojem se nalaze podaci koji se trebaju iscrtavati te se kamera postavlja tako da prati podatke koji se učitavaju. Cesium-ov sat (engl. *Clock*) se koristi za kontroliranje animacije, a pokretanje animacije se vrši kada je svojstvo *clock.shouldAnimate* postavljeno na *true*.

```
viewer.dataSources.add(
	Cesium.KmlDataSource.load('../bikeRide.kml')).then(
		function(dataSource)
		{
			viewer.clock.shouldAnimate = false;
			var rider = dataSource.entities.getById('tour');
			viewer.flyTo(rider).then(function()
			{
				viewer.trackedEntity = rider;
				viewer.selectedEntity = true;
				viewer.clock.shouldAnimate = true;
			});
		});
```

3.2.3. Prikaz na plohi i prikaz na sferi

Prikaz terena u Cesiumu, kao što je već ranije spomenuto, moguć je na tri načina: 2D (ploha) pogled, 3D (sfera) pogled i *Columbus* (ploha) pogled. Svi pogledi omogućuju zumiranje i rotiranje objekta na kojem se podaci iscrtavaju. 2D pogled ograničen je na okomiti pogled na plohu, dok je u *Columbus* i 3D pogledu moguće mijenjati kut pogleda.

Za prikaz 3D geometrije terena i slojeva u Cesiumu na plohi treba koristiti *Columbus* pogled. Ako se koristi 2D pogled onemogućen je prikaz geometrije u 3D. Da bi se moglo mijenjati prikaz s plohe u sferu i obrnuto koristi se funkcija transformacije.

Transformaciju scene, u željeni pogled, moguće je ostvariti pozivanjem sljedećih funkcije *scene.morphToColumbusView*(), *scene.morphTo2d*() ili *scene.morphTo3D().* Prilikom obavljanja transformacije, scena se zamrzava dok se transformacija ne obavi. Transformacija projekcijskog volumena vrši se na temelju matrice transformacije za pojedini pogled, te pozicije kamere.

3.2.4. Upravljanje slojevima

Slojevi koji se prikazuju na karti dijele se u nekoliko skupina: teren, bazna podloga, dodatni slojevi i izvori podataka. Teren je u *Cesium Viewer*-u objekt koji pripada sceni. Slojevi, bazni i dodatni, se nalaze u kolekciji slikovnih slojeva (*ImageryLayers*) kao dio scene, a izvori podataka se nalaze u kolekciji izvora podataka (*DataSources*) koja je objekt u *Viewer-u*.

Dobivanje te načini prikazivanja terena opisani su u poglavlju 3.1. Ako se dogodi greška i geometrija terena se ne može učitati, svi slojevi će se prikazivati na ravnoj podlozi.

Bazna podloga označava sloj koji se dodaje direktno na površinu terena. Bazni sloj se u kolekciji nalazi na prvoj poziciji. Radi boljih karakteristika iscrtavanja terena i slojeva, prilikom zamjene baznog sloja, potrebno je stari sloj maknuti iz kolekcije te zamijeniti novim. Isječak koda prikazuje funkciju koja postavlja novi bazni sloj.

```
function changeBaseLayer(viewer, name, imageryProvider)
{
	var new_layer;
	if ( viewer.scene.imageryLayers.length !== 0) {
		var old_layer = widget.scene.imageryLayers.get(0);
		widget.scene.imageryLayers.remove(old_layer, true);
		new_layer = new Cesium.ImageryLayer(imageryProvider);
	}
	new_layer.name = name;
	widget.scene.imageryLayers.add(new_layer,0);
}
```

Slikovni slojevi, koji ne čine baznu podlogu, dodaju se funkcijom *imageryLayers.add*() na kraj kolekcije slikovnih slojeva. Slojevi se na scenu iscrtavaju redoslijedom kojim su nalaze u kolekciji. Postoji nekoliko funkcija kojima se može upravljati redoslijedom slikovnih slojeva:

- *lower(layer)* - spušta sloj za jednu poziciju u kolekciji
- *raise(layer)* - podiže sloj za jednu poziciju u kolekciji
- *lowerToBottom* - postavlja sloj na dno kolekcije
- *raiseToTop(layer)* - postavlja sloj na vrh kolekcije

Sljedeći isječak koda prikazuje funkciju koja dodaje novi sloj u kolekciju ovisno o parametrima koje primi. U *widgetu* se nalazi scena, *name* predstavlja ima sloja koji se dodaje, a iz *imageryProvidera* se instancira novi slikovni sloj. Parametar *alpha* označava prozirnost sloja, a ako je parametar *show* jednak *true* sloj će biti vidljiv. Ova funkcija se poziva i kada se želi maknuti sloj iz scene, a tada parametar *show* treba imati vrijednost *false*.

```
function addNewImLayer(widget, name, imageryProvider, alpha, show) {
	var layer = new Cesium.ImageryLayer(imageryProvider);
	if (widget.scene.imageryLayers.contains(layer)) {
		var existing_layer = widget.scene.imageryLayers.get(layer);
		existing_layer.alpha = alpha;
		existing_layer.show = show;
	}
	else {
		widget.scene.imageryLayers.add(layer);
		layer.show = show;
		layer.alpha = alpha;
		layer.name = name;
	}
 }
```

Dodavanje novog izvora podataka obavlja se funkcijom *addNewDataSource*() koja je prikazana u sljedećem isječku koda. Parametar *remove* ovisno o tome ima li vrijednost *true* ili *false* označava da li *source* treba dodati ili maknuti iz kolekcije izvora podataka. Parametar *dataSource* koji se dodaje u kolekciju izvora podataka označava KML, GeoJson, TopoJson ili CZML izvor podataka koji je stvoren odgovarajućim konstruktorom. Također, u isječku se nalazi primjer poziva funkcije s KML izvorom podataka koji predstavlja turu jedrenja od Šibenika prema Skradinu. Slika 3-5 prikazuje izvor podataka dodan nakon učitavanja terena i baznog sloja.

```
function addNewDataSource(widget, dataSource, remove) {
        if (remove === true && widget.dataSources.contains(dataSource)) {
                var ds = widget.dataSources.get(dataSource);
                widget.remove(ds, remove);
        }
        if (remove === false && !(widget.dataSources.contains(dataSource))){
                widget.dataSources.add(dataSource);
        }
        var kml_data = Cesium.KmlDataSource.load('Sibenik2Skra.kmz'));
        addNewDataSource(viewer, kml_data, false);
```

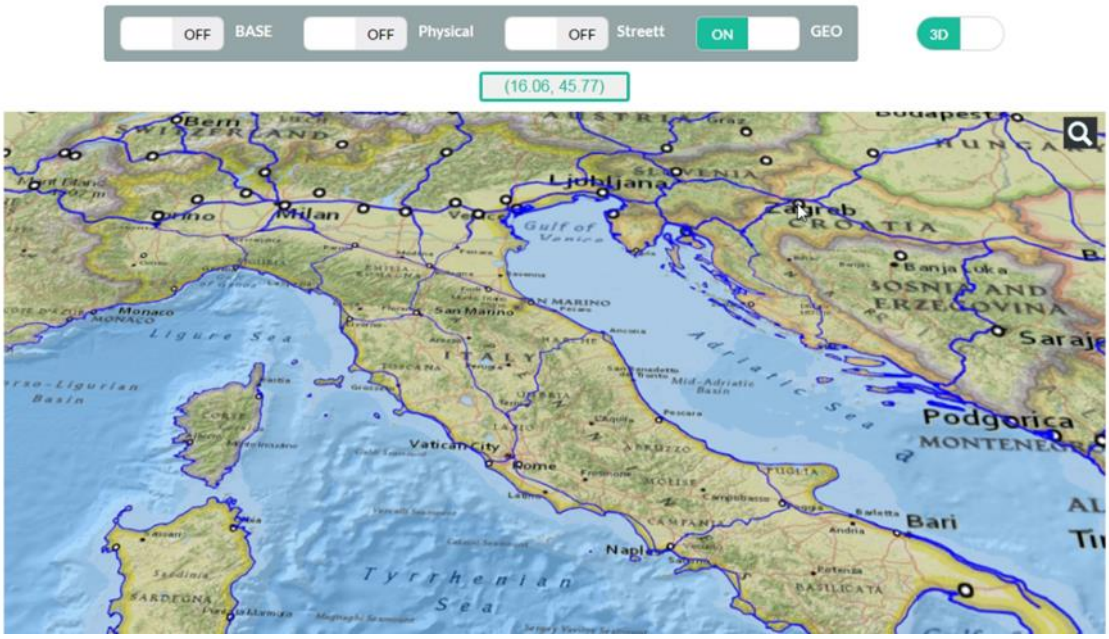
Slika 3-5 Učitan KML izvor podataka nakon dodavanja terena i baznog sloja

3.2.5. Prikaz koordinata nad kojima je miš

ScreenSpaceEventHandler() je objekt u Cesium-u koji upravlja događajima na sceni koja se prikazuje. Pomoću funkcije *getInputAction*() definira se tip događaja koji se prati te objekt nad kojim se prate događaji. Sljedeći isječak koda odgovara dohvaćanju koordinata i njihovom ispisu na objekt čiji identifikator ima vrijednost jednaku vrijednosti parametra *input_id*.

```
function start_picking_handler(ellipsoid, widget, input_id){
       var handler = new Cesium.ScreenSpaceEventHandler  (widget.scene.canvas);
       handler.setInputAction(function(movement) {
              var cartesian = widget.camera.pickEllipsoid
                                    (movement.endPosition, ellipsoid);
       if (cartesian) {
            var cartographic = ellipsoid.cartesianToCartographic(cartesian);
            var longitudeString = Cesium.Math.toDegrees(cartographic.longitude);
            var latitudeString = Cesium.Math.toDegrees(cartographic.latitude);
           var text = '(' + longitudeString + ', ' + latitudeString + ')';
           $(input_id).val(text);
       } else {
             $(input_id).val("");
       }
    }, Cesium.ScreenSpaceEventType.MOUSE_MOVE);
}
```

U ovom slučaju, kada se želi pratiti koordinate nad kojima je miš, tip događaja je *movement*. Budući da se koordinate žele ispisati, potrebno ih je dohvatiti. Dohvaća se pozicija koja odgovara krajnjem položaju pokreta miša na sferi. Koordinate su zadane u kartezijevom sustavu te ih se pretvara u geografsku dužinu i širinu. Prikaz koordinata u ovisnosti o poziciji miša prikazuje Slika 3-6.



Slika 3-6 Prikaz koordinata nad kojima se nalazi miš

4. VIŠESLOJNI INFORMACIJSKI SUSTAV

Prikaz višeslojnog sustava podijeljen je na dva dijela, jedan se fokusira na prostor Hrvatske, a drugi za svijet. Oba prikaza funkcioniraju na isti način, ali uključuju različite podatke za prikaz. Teren Hrvatske učitava se u formatu *heightmap-1.0*, a stvoren je na način prikazan u poglavlju 3.1.3. Drugi teren koji je korišten za svijet je *STK World Terrain*.

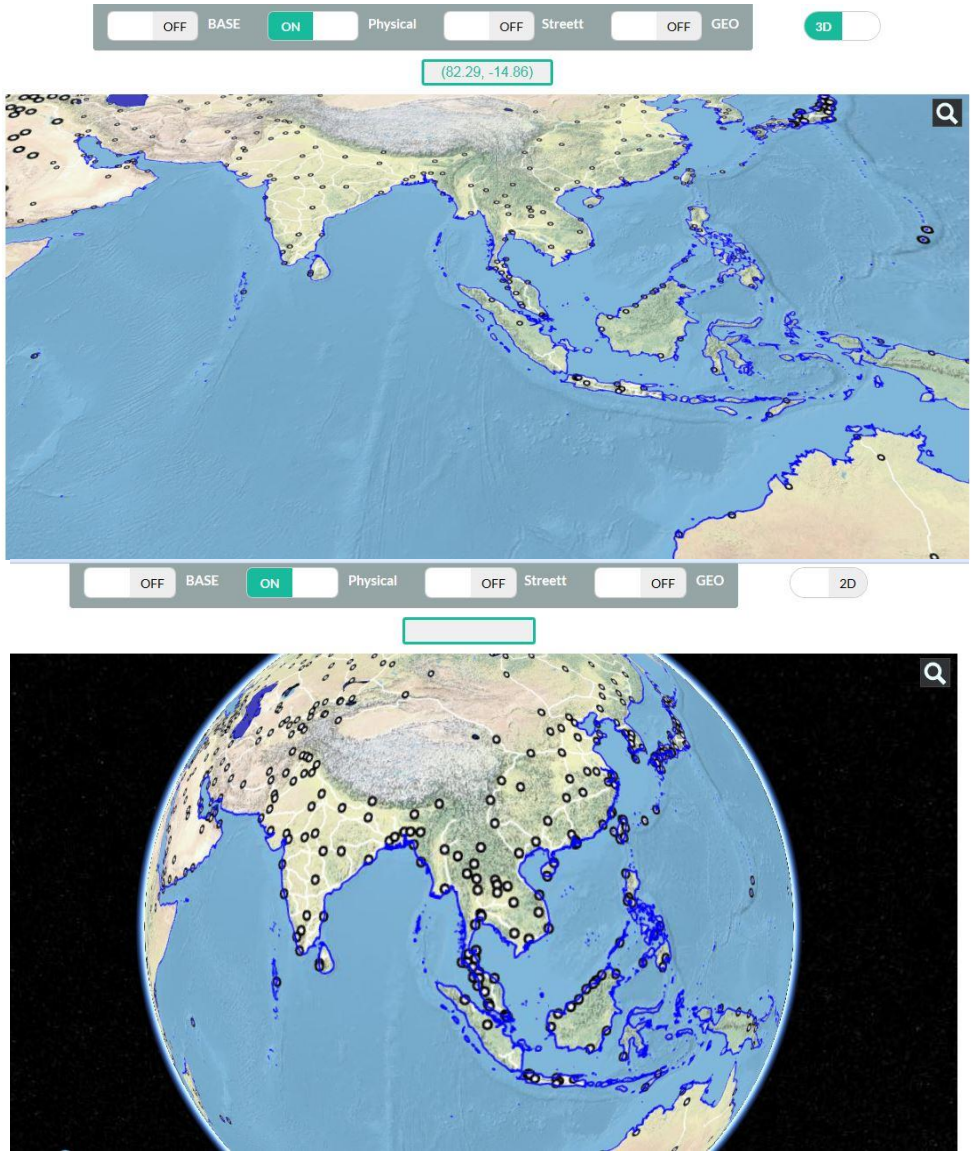
Iznad prikaza karte nalazi se izbornik baznih slojeva. Slika 4-1 prikazuje opcije promjene baznih slojeva za Hrvatsku, a Slika 4-2 za svijet. Oznaka „ON“ označava da se taj sloj trenutno nalazi na karti kao bazni sloj. Za promjenu baznog sloja potrebno je kliknuti lijevom tipkom miša na dio izbornika koji odgovara tom sloju i obaviti će se zamjena baznog sloja na način opisan u poglavlju 3.2.4. Desno od izbornika baznih slojeva nalazi se izbornik za prebacivanje pogleda sa sfere na plohu i obratno.

Slika 4-1 Opcije za prikaz na karti - Hrvatska

Slika 4-2 Opcije za prikaz na karti - Svijet

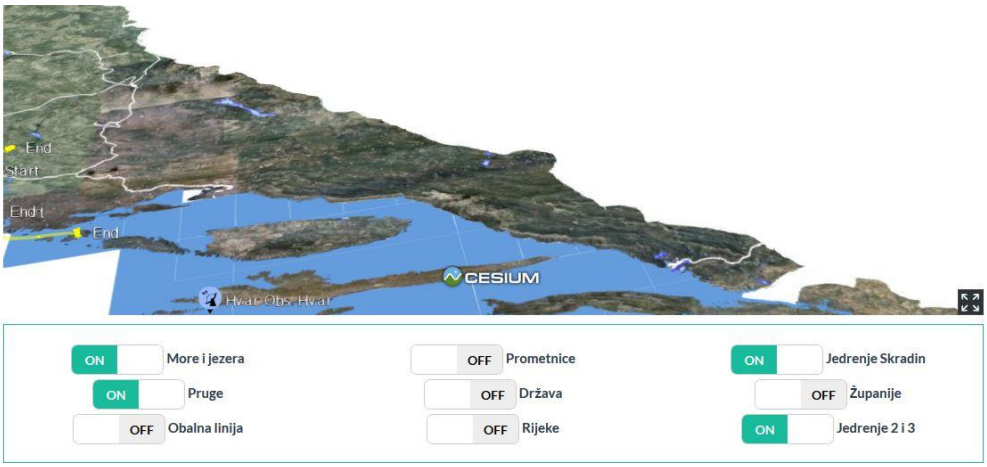
Bazni slojevi se dohvaćaju s poslužitelja WMS zahtjevima opisanim u poglavlju 3.2.1. Bazni slojevi koji se koriste za Hrvatsku su DOF (Digitalni ortofolio), HOK (Hrvatska osnovna karta) i TK25 (Topografska karta), a javno su dostupni na stranici geoportala [8] te GEO sloj s ArcGIS poslužitelja [2] u bazi *NatGeo_World_Map*. Svi bazni slojevi na karti svijeta dohvaćaju se s ArcGIS poslužitelja, a nalaze se u različitim bazama. Sloj BASE nalazi se u *World_Imagery* i jednak je sloju koju Cesium postavlja na kartu, ako ništa drugo nije zadano. Sloj *Physical* iz baze *World_Physical_Map* predstavlja fizičku kartu svijeta, a sloj *Street* iz baze *World_Street_Map* kartu koja u prikazu uključuje ulice. Usporedba prikaza na plohi i na sferi za kartu svijeta uz baznu podlogu *Physical*

prikazuje Slika 4-3. Također, na slici se može vidjeti da su uključeni i dodatni slojevi koji prikazuju obalnu liniju, pruge i pozicije zračnih luka u svijetu. Ispod ovih izbornika nalazi se ispis trenutnih koordinata u ovisnosti o položaju miša, što je, također, vidljivo na slici.

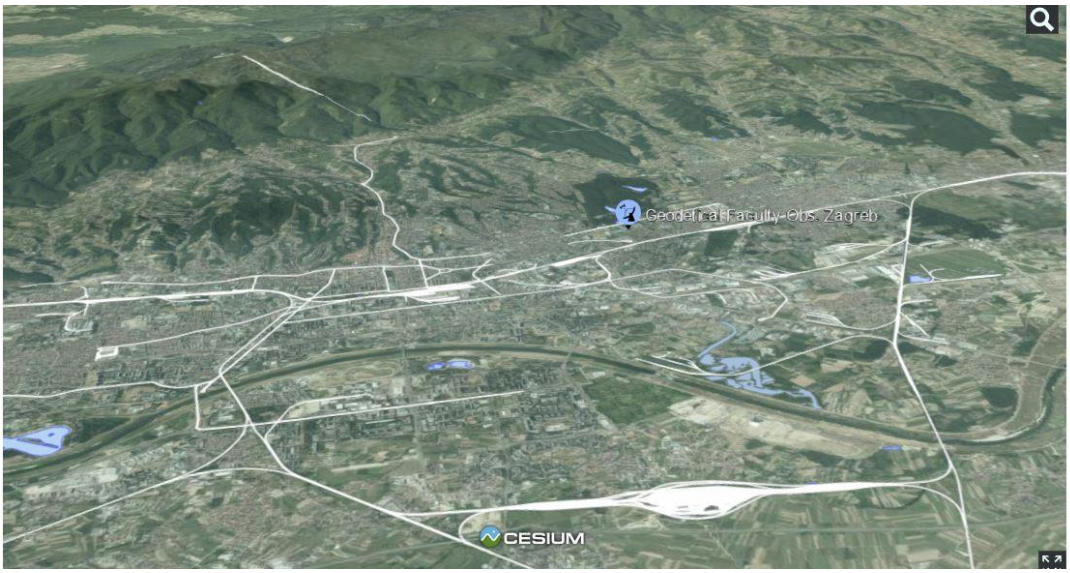


Slika 4-3 Prikaz na plohi i na sferi

Ispod karte se nalazi se izbornik koji omogućuje postavljanje aktivnosti slojeva. Slika 4-4 prikazuje izgled izbornika dodatnih slojeva i izvora podataka te njihov prikaz na karti Hrvatske s odabranim dodatnim slojevima. Slika 4-5 također prikazuje kartu Hrvatske sa uključenih nekoliko slojeva i izvorom podataka.



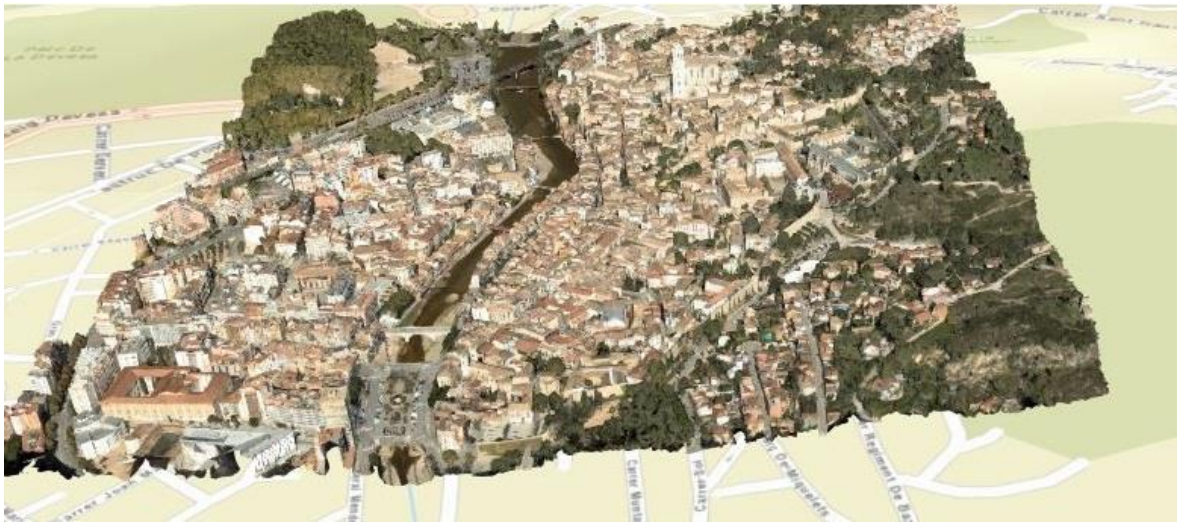
Slika 4-4 Odabir i prikaz dodatnih slojeva na karti Hrvatske



Slika 4-5 Prikaz terena Zagreba slojeve Pruge, Jezera i izvor podataka Zvjezdarnice

Slika 4-6 prikazuje učitani 3D model grada na karti Svijeta. Grad je učitan iz czml izvora podataka, uz *Base* kao osnovni sloj. Slika 4-7 prikazuje grad uz uključene druge slojeve.

Slika 4-6 3D prikaz grada učitanog iz czml izvora podataka na baznoj podlogi terena karte Svijeta



Slika 4-7 Prikaz grada uz uključene druge slojeve

5. NEDOSTATCI I PRIJEDLOZI

Cesium nudi velik broj mogućnosti za prikaz geoinformacijskih podataka, ali dok je 2D prikaz prilično dobro obuhvaćen, 3D prikazi imaju dosta problema i nedostataka. Za prikaz geometrije terena Cesium podržava samo vlastite formate koji bi se trebali dohvaćati s njihovog poslužitelja. Korištenje tog poslužitelja nije besplatno, ali nudi mogućnost da se iz proizvoljne DEM datoteke učita i prikaže teren na sceni.

Ipak, DEM datoteka koja se može koristiti je ograničena. Jedini koordinatni sustav koji se podržava je WGS 84 (EPSG:4326). Bilo bi dobro kada bi Cesium uveo podršku za Proj.4 te na taj način omogućio reprojekciju koordinata. Kod učitavanja terena se događa da se teren ponekad učita, a ponekad ne učita geometriju, a jedina poruka ispisuje se u web konzoli u obliku „Failed to obtain image tile at X: 1 Y: 1 Z: 1“.

Dosta nedostataka i grešaka pri izvođenju događa se zbog nedovršenosti pojedinih dijelova. Naročito pri učitavanju 3D geometrije ili složenijih slojeva dolazi do izražaja sporost iscrtavanja te prilično često dolazi i do zamrzavanja web preglednika. Jedan od glavnih razloga je nedovršenost implementacije primitiva *QuadTreePrimitive*, a koji upravlja razinama detalja u prikazu. Njegova implementacija bi zasigurno trebala biti poboljšana prije nego li se krene s dodavanjem novih mogućnosti.

Iako je u dokumentaciji i opisu alata navedeno podosta formata koje Cesium podržava, tu se, također, pojavljuju problemi. U trenutku kada dolazi do potrebe za učitavanjem datoteka s podacima dolazi se do zaključka da je većina formata podržana samo djelomično. Na primjer, CZML, koji je Cesium-ov format za dinamičke objekte, je teško iskoristiv za vizualizaciju, primarno jer prilikom pojavljivanja neke greške nema apsolutno nikakve dojave o pograšci, a na sceni se ništa ne pokaže.

Jedini format u kojem se mogu učitavati 3D modeli, a podržan je i od drugih web grafičkih pogona, u Cesium je glTF. Cesium sam nudi sučelje za pretvorbu COLLADA modela u glTF format. I tu nastaju problemi. Pretvorba je uspješna, ali prilikom učitavanja modela u aplikaciju dolazi do pogreške prilikom vizualizacije bez dodatnih objašnjenja.

Izračun normala u Cesium-u trebalo bi ostvariti na nižoj razini da se može pozivati jednom funkcijom, npr. naziva *Globe.getGroundNormal()*. Zasada je normalnu moguće izračunati klijentski se na temelju 4 susjednih visinskih uzoraka.

Prilikom transformiranja pogleda scene između 2D, 3D i *Columbus* pogleda kamera se uvijek vraća u početno definirani položaj te se mora tražiti pozicija na kojoj je kamera prethodno bila.

S obzirom da je sama podrška za 3D prikaze dosta nestabilna, nedostaju alati koji bi omogućili analizu geometrije. Međutim, za neke jednostavne aplikacije i prikaze Cesium je dosta dobar, ali ne i za složenije 3D prikaze.

6. ZAKLJUČAK

Cesium je grafički pogon otvorenog koda, u razvoju, baziran na HTML5 standardu i WebGL-u, a usmjeren na prikaz različitih geoinformacijskih podataka. Njegova arhitektura čini ga prilagodljivim za dodavanje novih alata i nadogradnju postojećih.

Mogućnost prikaza podataka na karti moguća je na tri načina: 2D pogled (ploha), *Columbus* (zapravo 2.5D pogled) i 3D (sfera) pogled. Podrška za poslužitelje koji se i inače koriste u web aplikacijama koje rade s kartama je dobra. Prikaz 2D geometrije je dosta dobar, dok pri samom 3D prikazu ima nedostataka.

Podržani su različiti formati za učitavanje podataka, ali tome treba pristupiti s oprezom jer većina formata nije u potpunosti podržana. Komunikacija Cesium-a s Geoserverom je prilično dobra. Prikaz terena se da dosta dobro napraviti, a slojevi koji se dodaju na teren se dobro prilagođavaju terenu.

Ostvarenje sustava višeslojne vizualizacije u ovom radu pokazuje kako se s dosadašnjim razvojem Cesium-a mogu ostvariti zadovoljavajući rezultati i prikaz 3D geometrije, ali daje i određene naznake o problemima unutar pogona na kojima treba poraditi.

Početak rada i razvoja aplikacija u Cesium-u nije težak, ali za ostvarenje malo složenijih prikaza potrebno je ne samo iskustvo u WebGL-u (ili OpenGL-u), već i poznavanje rada s geoinformacijskim podacima. U konačnici, postoji obuhvatna dokumentacija o korištenju i implementaciji Cesium web grafičkog pogona. Iako, možda u ovom trenutku nije u potpunosti podoban za složene 3D vizualizacije, postavljeni su temelji koji mu mogu omogućiti da to postane.

LITERATURA

[1] Patrick Cozzi, *Cesium Architecture*, 15.5.2015., https://github.com/AnalyticalGraphicsInc/cesium/wiki/Architecture, 16.3.2015.

[2] *ArcGIS REST Services Directory*, http://server.arcgisonline.com/arcgis/rest/services/, 20.3.2015.

[3] Tile Map Service Specification, http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification, 15.4.2015.

[4] *ASTER Global Digital Elevation Map Announcement,* 17.10.2011., http://asterweb.jpl.nasa.gov/gdem.asp, 17.4.2015.

[5] *SRTM 90m Digital Elevation Data*, 19.8.2008., http://srtm.csi.cgiar.org/, 20.5.2015.

[6] *DEM (Digital Elevation Models) files*, http://vterrain.org/Elevation/dem.html, 22.5.2015.

[7] *Geoportal – Podaci i servisi,* http://geoportal.dgu.hr/podaci-i-servisi/, 22.5.2015.

[8] Homme Zwaagstra, *Cesium Terrain Builder,* 20.5.2015., htttp://github.com/geo-data/cesium-terrain-builder, 30.5.2015.

[9] *Cesium Tutorials*, http://cesiumjs.org/tutorials.html, 3.6.2015.

[10] Bjørn Sandvik, *Creating 3D terrains with Cesium*, 12.10.2014., http://blog.thematicmapping.org/2014/10/3d-terrains-with-cesium.hml, 30.5.2015.

[11] *Geoserver Documentation,* http://docs.geoserver.org/, 3.6.2015.

[12] *Web Map Service (WMS)* http://girona-geoserverworkshop.readthedocs.org/en/latest/overview/wms.html, 25.6.2015.

[13] Patrick Cozzi, *Graphics Tech in Cesium – The Graphical Stack*, 26.5.2015., http://cesiumjs.org/2015/05/26/Graphics-Tech-in-Cesium-Stack/, 27.6.2015.

[14] Patrick Cozzi, *Graphics Tech in Cesium – Renderer Architecture*, 15.5.2015, http://cesiumjs.org/2015/05/15/Graphics-Tech-in-Cesium-Architecture/, 27.6.2015.

[15] Patrick Cozzi, *Graphics Tech in Cesium – Rendering a frame*, 14.5.2015., http://cesiumjs.org/2015/05/14/Graphics-Tech-in-Cesium/, 27.6.2015.

[16] *Cesium Documentation*, https://cesiumjs.org/Cesium/Build/Documentation/index.html, 28.6.2015.

SAŽETAK

U radu je dan pregled arhitekture web grafičkog pogona Cesium i mogućnosti prikaza kroz niz primjera. Cesium je Javascript biblioteka otvorenog koda za prikaz geoinformacijskih podataka u web preglednicima, bez korištenja dodataka. U Cesiumu je ostvaren sustav koji omogućuje 3D vizualizaciju višeslojnih geinformacijskih podataka. Vizualizacija omogućuje uključivanje i isključivanje pojedinih slojeva uz osiguranu konzistentnost i ispravnu povezanost. Ostvarena je komunikacija sa standardnim poslužiteljima geoinformacijskih podataka, na primjer, *Geoserver* i *ArcGisMapServer*, te učitavanje podataka iz odgovarajućih izvora podataka. Navedene su prednosti i nedostaci ovog grafičkog pogona za navedenu primjenu te prikazani rezultati vizualizacija u Cesium-u.

Ključne riječi: Cesium, JavaScript, WebGL, GeoServer, geoinformacijski podaci, 3D višeslojna vizualizacija

ABSTRACT

This paper presents an overview of the architecture of web graphical engine Cesium and displays its possibilities through a series of examples. Cesium is an open source Javascript library for displaying geoinformation data in web browsers without requiring any extra add-ons. The paper describes the implementation of 3D multi-layer visualization of geoinformation data. Visualization provides the possibility of showing and removing individual layers as well as ensuring consistency. Standard geospatial servers, such as Geoserver and ArcGisMapServer are used for data serving. Also, data for visualization is loaded from supported data sources. Advantages and disadventages of Cesium engine for this use are discussed and visualization results presented.

Keywords: Cesium, JavaScript, WebGL, GeoServer, geoinformation data, 3D multi-layer visualizations

DODATAK

Upute za instalaciju i pokretanje

Cesium je izgrađen na novim HTML5 tehnologijama, i projekte je nužno pokretati u preglednicima koji podržavaju WebGL, npr. najnovije verzije Google Chrome-a, Mozzila Firefox-a, Opere i Safari-ja. Poželjno je osvježiti status upravljačkih programa (engl. *drivers*) video/grafičke kartice kako bi podrška za prikaz 3D bila što bolja.

Najjednostavnije razvojno okruženje za Cesium je Notepad++. Sa stranice *https://cesiumjs.org/downloads.html* potrbeno je skinuti željenu verziju Cesiuma i spremiti ju na disk. Root direktorij od Cesium-a treba sadržavati sljedeće direktorije: Apps, Build, Source, Specs i ThirdParty, te sljedeće datoteke: CHANGES.md, favicon.ico, index.html, LICENSE.md, logo.png, package.json, README.md i server.js.

Kako bi se Cesium aplikacija pokrenula potreban je lokalni web poslužitelj na kojem će podaci biti postavljeni. WampServer poslužitelj je dobar izbor, a može se skinuti sa sljedeće poveznice: *http://www.wampserver.com/en/*.

Nakon instalacije, potrebno je u www direktorij na WampServeru kopirati *root* direktorij Cesiuma. Nakon toga se pokrene WampServer te se provjeri uspješnosti pokretanja Cesium-a upisivanjem *"http://localhost/Apps/HelloWorld.html"* te se otvara aplikacija u pregledniku.

Potrebno je instalirati Geoserver koji je dostupan na sljedećoj poveznici *http://geoserver.org/download/*. Preporučne verzija, zbog standardiziranih formata, za pokretanje ovog rada je 2.6.3.

Dodatak koji je potreban za određene vizualizacije s Geoserverom je *Cesium-GeoserverTerrainPro*vider, a dostupan na stranici *https://github.com/kaktus40/Cesium-GeoserverTerrainProvider*. Sve direktorije i mape potrebno je spremiti u direktorij unutar root direktorija Cesium-a, a taj direktorij imenovati *Cesium-GeoserverTerrainProvider-master*.

Kako bi se omogućili CORS zahtjevi (engl. *Cross-Origin Resource Sharing*) potrebno je na poslužitelju izmijeniti *httpd.conf datoteku* i dodati sljedeći sadržaj:

```
<Proxy /geoserver>
        Require all granted
</Proxy>
ProxyPass /geoserver http://localhost:8001/geoserver
ProxyPassRevers /geoserver http://localhost:8001/geoserver
```

Port 8001 potrebno je zamijeniti odgovarajućim portom na kojem je geoserver smješten. Također, kako bi se omogućili WMS zahtjevi potrebno je izmijeniti datoteku *httpd-vhosts.conf* te dodati sljedeći sadržaj:

```
<VirtualHost *:80>
<Location \wms>
        Odrer allow, deny
        Allow from all
                ProxyPass /geoserver http://localhost:8001/geoserver/wms
                ProxyPassRevers /geoserver http://localhost:8001/geoserver/wms

</Location>

</VirtualHost>
```

Za postavljanje sustava za višeslojnu vizualizaciju potrebno je direktorij *FINAL* iz priloga potrebno je smjestiti u *Cesium/Apps* direktorij na serveru. Direktorij *FINAL* sadrži sljedeće direktorije: *css, font-awesome, fonts, img, js, less* te datoteka *indeks.html*. Upisivanjem u preglednik adrese *localhost/Apps/FINAL/indeks.html* otvorit će se početna stranica razvijenog sustava.

Datoteke u direktoriju *shp* potrebno je postaviti na Geoserver. Datoteke iz direktorija *ws_hrvatska* postavljaju se na Geoserver u *workspace ws_hrvatska* te se svakom sloju dodaje istoimena stilska datoteka, a datoteke iz direktorija *ws_svijet* se postavljaju u workspace *ws_svijet* prema uputama koje se nalaze na sljedećoj poveznici *http://docs.geoserver.org/2.6.x/en/user/gettingstarted/shapefile-quickstart/index.html.*