Eötvös Loránd Tudományegyetem

Informatikai Kar

Térképtudományi és Geoinformatikai Tanszék

Tematikus térképek készítése a D3.JS függvénykönyvtár segítségével

Készítette:

Rátvai Dániel

Térképész mesterszakos hallgató

Témavezető:

Ungvári Zsuzsanna

adjunktus

Budapest, 2018

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

TÉRKÉPTUDOMÁNYI ÉS GEOINFORMATIKAI TANSZÉK

DIPLOMAMUNKA-TÉMA BEJELENTŐ

Név: Rátvai Dániel

Neptun kód: YBTKVG

Szak: **térképész MSc**

Témavezető neve: Ungvári Zsuzsanna

munkahelyének neve és címe:

beosztása és iskolai végzettsége:

A dolgozat címe:

Tematikus térképek készítése a D3.JS függvénykönyvtár segítségével

A dolgozat témája:

A D3 JavaScript függvénykönyvtár felhasználási lehetőségeinek bemutatása különböző tematikus térképek segítségével. Az elkészült térképek minőségének összehasonlítása QGIS-ben készült térképekkel.

A témavezetést vállalom.

.....

(a témavezető aláírása)

Kérem a diplomamunka témájának jóváhagyását.

Budapest, 2017.05.16.

.....

(a hallgató aláírása)

A diplomamunka-témát az Informatikai Kar jóváhagyta.

Budapest, 20……

……..

(témát engedélyező tanszék vezetője)

Tartalomjegyzék

1. Bevezetés

Dolgozatom témájaként a d3 JavaScript függvénykönyvtár webtérképészeti lehetőségeinek bemutatását tűztem ki célul.

Napjainkban az internet az emberek mindennapjainak részévé vált, a felhasználók száma az okostelefonok, illetve a táblagépek elterjedésével egyre csak növekszik. Ez a tendencia a térképészetre is kifejti a hatását, ugyanis egy weboldalba ágyazott térkép sokkal nagyobb közönséghez jut el, mint a nyomtatott verziója. Hatalmas előny, hogy a weben megjelenő térképeknél lehetőségünk van egy interaktív felületet létrehozni, amellyel sokkal több adatot tudunk megjeleníteni, ezáltal az olvasó számára is „izgalmasabbá" tudjuk tenni az elkészített művünket.

Számos elterjedt és népszerű megoldás létezik erre a feladatra, azonban a dolgozatomban egy kevésbé elterjedt módszert szeretnék ismertetni.

A d3.JS egy JavaScript nyelven íródott függvénykönyvtár, mely segítségével dinamikus adatmegjelenítést lehet elérni. Mint minden weboldalszerkesztésre alkalmas nyelvnek, ennek is az alapját a HTML programozási nyelv jelenti. A vektorgrafikus alakzatok megjelenítése az SVG leíró nyelv segítségével történik, míg az objektumok stílusának definiálására CSS-t használ.

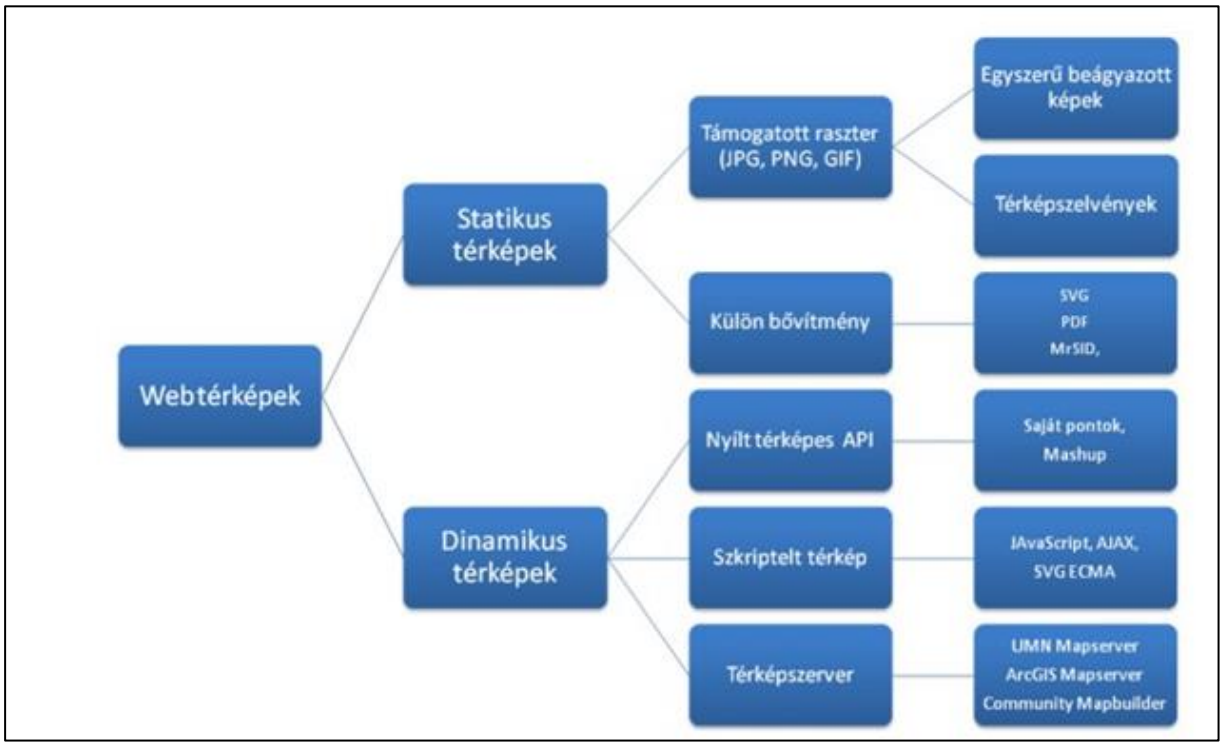
Jól látható, hogy egy d3 állomány elkészítéséhez HTML, SVG, CSS legalább alapszintű ismerete elengedhetetlen, ugyanakkor a JavaScript nyelv ismeretének a hiánya – alapszinten – nem jelent gondot az állományok elkészítésében.

Dolgozatomban ismertetem a d3 függvénykönyvtár felhasználási lehetőségeit. Röviden bemutatom az általa használt nyelveket, fájlformátumokat és a használni kívánt keretprogramokat. Ezeken kívül bemutatom a QGIS program d3 Map Renderer nevű modulját, illetve ezzel a térinformatikai programmal térképeket készítek, amelyeket utána d3.js segítségével interaktív webtérképpé alakítok. A kész térképeket összehasonlítom, majd a dolgozat végén összegzem a d3.js felhasználási lehetőségeit.

2. Webtérképészet

Régen a térképkészítés kizárólag a térképészek feladata volt, de a térinformatika és a számítógépes grafika fejlődésével ez megváltozott. A web elterjedéséhez kapcsolódó információs forradalom tovább növelte azok számát, akik térképeket készítenek és publikálnak. A legújabb tendencia az, hogy már nem elsősorban a térbeli adatok szolgáltatói alakítják ki a webtérképek „dizájnját", hanem maguk a felhasználók is képesek bizonyos beállítások segítségével a saját ízlésünknek megfelelően áttervezni a megjelenő térképet.

A webtérképeknek két nagy típusát különítik el a szakirodalomban: a statikus és a dinamikus térképeket. Ezen belül további megoldások is megfigyelhetők. A tendencia egyértelműen azt mutatja, hogy a professzionális alkalmazások területén kizárólag a dinamikus térképek az elfogadottak. Mind a statikus, mind a dinamikus térképek webes publikálásának megtalálhatjuk a speciális eseteit, melyet a következő sematikus ábrán mutatom be [Pődör, 2010].



1. ábra: A webtérképek publikálásának főbb esetei (Guszlev Antal nyomán)[1]

[1] Az ábra elérése: http://www.tankonyvtar.hu/hu/tartalom/tamop425/0027_KAR7/ch01s04.html

Statikus térképek megjelenítésére a legelterjedtebb megoldás, ha valamilyen raszteres állományként (JPG, PNG stb.) beágyazzuk a HTML oldalba.

A statikus térképek esetén is lehetséges interaktív funkciók alkalmazása. Ilyen például, amikor a térképet mozgatni, kicsinyíteni és nagyítani tudjuk, továbbá hivatkozás is hozzáadható. Ezek az image map-ek, a térbeli adatok csatolóiként is működhetnek. Leggyakrabban különféle cikkekben használják.

A dinamikus térképek legfontosabb jellemzője a változás és változtatás lehetősége, továbbá hogy támogatják a különböző multimédiás megjelenítési formákat. Ennek legegyszerűbb módja az animáció, amikor egymás után különböző térképeket vetítünk le, melyek valamilyen változást mutatnak be (pl. lakosság növekedése, egy város területi kiterjedésének változása) és ezeket a képeket összefűzzük. Ez kifejezetten jellemző a meteorológiai szervezetek weboldalaira [Pődör, 2010].

A dinamikus térképek egyik leginkább kedvelt változata a „szkriptelt" térképek alkalmazása, melyek valamilyen szkript nyelv használata segítségével készülnek. Dolgozatomban ilyen térképek elkészítését tűztem ki célul.

Dinamikus térinformatikai adatokon alapuló térképek jellemzői:

- Térképszerverek szolgáltatják a tartalmat
- Dinamikus tartalom érhető el
- Háttér adatbázissal rendelkezik
- Sokféle térképi funkció elérhető az egyes alkalmazásokon belül
- HTML, JavaScript, AJAX, Java, ASP, ASP.NET, PHP nyelvek közül valamelyikkel programozható

Google Maps

A Google Maps segítségével képesek vagyunk nagyon egyszerű térképeket készíteni és beágyazni egy adott honlapba úgy, hogy alaptérképként a Google Maps-et használjuk fel.

A Google Maps lehetővé teszi számunkra, hogy adatainkat, élményeinket, melyek helyhez kötöttek egyszerűen és jelenleg ingyenesen egy saját térképen tároljuk és jelenítsük meg. Ha jól sikerült a térképünk vagy az esetleg mások számára is érdekes lehet, akkor nyilvánossá

tehetjük, így megjelenik az interneten. A Google MyMaps-nek (saját térkép) nevezi ezt a szolgáltatást.

Google Fusion

A Google Fusion Tables a Google saját fejlesztése, tehát saját háttértérképet, a Google Maps térképes, illetve műholdas megjelenítését használja, melyre pontokat, vonalakat vagy felületeket képes felszerelni. Névrajz nem jelenik meg a térképen. Szűrést is végezhetünk a térképen a tábla adatai alapján. Segítségével egyszerűbb tematikus térképek állíthatóak elő [Google, 2013].

OpenLayers

Az OpenLayers egy nyílt forráskódú kliens oldali JavaScript függvénykönyvtár, melynek segítségével különféle forrásokból származó térképi adatok jeleníthetők meg interaktív térképként weboldalakon. Az OpenLayers első változatát a MetaCarta nevű amerikai cég fejlesztette ki 2006-ban. 2007 novemberétől a projektet átvette az Open Source Geospatial Foundation (OSGeo). Az OpenLayers csak az adatok megjelenítését végzi, ezért mindenképpen szükség van a háttérben valamilyen adatforrásra. Ez utóbbi rendkívül sokféle lehet: egy egyszerű raszteres képtől kezdve az OpenStreetMap térképén és a különféle WMS, TMS szervereken keresztül különféle vektoros adatszolgáltatókig rengeteg lehetőség akad [Gede, 2012].

OpenStreetMap (OSM)

Az OpenStreetMap (OSM) egy közösségi csapatmunkán alapuló projekt, aminek célja egy szabadon szerkeszthető és felhasználható térkép készítése az egész világról. A térképek hordozható GPS eszközökből, légi fotókból és egyéb szabad forrásokból származó adatok, vagy egyszerű helyismeret alapján készültek. A regisztrált felhasználók feltölthetnek GPS nyomvonalakat, és szerkeszthetik a vektoros adatokat is adott szerkesztőeszközök használatával. A Google Maps-hez hasonlóan az OSM-et is felhasználhatjuk alaptérképnek. Több térképkészítő oldal is felhasználja alapként.

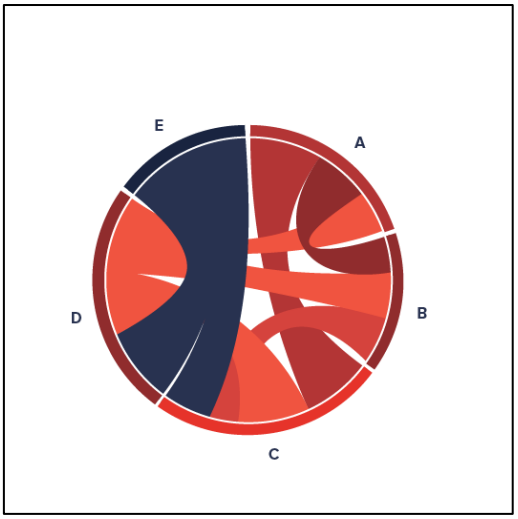
3. d3 JavaScript függvénykönyvtár

Létrejötte

A d3 JavaScript (Data Driven Documents) könyvtárat Mike Bostock írta 2011-ben. 2014-ig a New York Times adatmegjelenítési osztályánál dolgozott vezető beosztásban, ahol az újság weboldalán[2] történő különböző adatbázisokon alapuló adatmegjelenítés volt a feladata. A hosszú évek alatt a saját munkája megkönnyítése érdekében megírta a d3 JavaScript függvénykönyvtárát. Az elkészült mű annyira jól sikerült, hogy Edward Tufte az Amerikai Yale egyetem vezető professzora azt mondta Bostockról, hogy *„…ő lesz az elkövetkező évek legfontosabb embere, hogy ha adat megjelenítésről beszélünk."* illetve a d3.js-ről pedig azt, hogy *„…ez a legújabb és a legerősebb web-függő adatmegjelenítési módszer napjainkban"* [Cookson, 2013].

Felhasználása

A d3.JS-t egy adatbázison alapuló adatmegjelenítésre kifejlesztett függvénykönyvtár. Elsősorban különböző típusú interaktív grafikonok létrehozására lett kifejlesztve. Különlegessége, hogy a legösszetettebb grafikonokat is viszonylag könnyen lehet elkészíteni vele, úgymint a chord-diagramot („ívelt húrok diagramja"). Mivel a térképeket is meglévő adatbázisból tudjuk létrehozni, így a d3 JavaScript függvénykönyvtár másodlagosan alkalmas arra is, hogy különféle tematikus térképeket hozzunk létre [Qi Zhu, 2013].



2. ábra: A chord-diagram

[2] https://www.nytimes.com/

Varázsa abban rejlik, hogy az eddig meglévő és jól működő web-dizájn elemeket összefogja és a JavaScript lehetőségeit kihasználva egy könnyen kezelhető fejlesztőkörnyezetet hoz létre, amellyel az adat megjelenítési formáinak csak a kreativitás szab határt. A hivatalos weboldalon[3] számos alkotás megtekinthető. A weboldalon a különböző grafikonon és térképeken kívül lehetőségünk van az egyes művekhez tartozó kódokat is megtekinteni. Található még egy összefoglalás a d3.JS működéséről, illetve innen lehet letölteni a függvénykönyvtárat. A letöltés, a dokumentumok, és a példák is teljesen ingyenesek, bárki szabadon felhasználhatja őket.

A d3.js legújabb verziója a 4-es, de jelenleg a legtöbb leírás a 3-as verzióhoz készült, ezért a dolgozatom elkészítéséhez ezt használtam fel.

3. ábra: A d3.JS hivatalos weboldalának kezdőoldala

[3] https://d3js.org/

4. Felhasznált programozási nyelvek és formátumok

HTML (*HyperText Markup Language)*

A HTML egy lapleírónyelv, amivel weboldalakat tudunk készíteni. Több verziója is elkészült az elmúlt években, jelenleg a HTML5 jelölésű verzió a legújabb. Legnagyobb változás a 4-es verzióhoz képest, hogy sikerült megoldani azt, hogy a különféle webes alkalmazásokhoz ne legyen szükség semmilyen plugin-nek (például az Adobe Flash) a telepítésére [Huszár, 2009].

Magát a leíró nyelvet nagyon hosszú lenne bemutatni, ezért csak néhány olyan alap tulajdonságot ismertetek, amely elengedhetetlen a dolgozatom megértéséhez.

A HTML valójában három nyelvi elemből építkezik: vannak benne vezérlő szerkezetek (angolul: tag) és attribútumok. Minden HTML fájlnak tartalmaznia kell a következőket:

- <html lang="en">…</html>: Az egész kódsorunkat közre fogja. Ezzel kezdődik a kód és ezzel zárul. A ’lang=”en”’ rész az oldal nyelvét adja meg. Ezt nem kötelező megadni, de használata előnyös, mivel például a Google keresésnél beállítható, hogy csak a magyar (’=hu’) oldalak között keressen.
- <head>...</head>: Ez a dokumentum fejléce, amiket ide írunk, az közvetlenül nem jelenik meg a weboldalon, de fontos adatokat tartalmaz a dokumentum egészére vonatkozólag. (pl.: karakterkódolás, weboldal címe)
- <body>...</body>: Ez a vezérlő szerkezet tartalmazza a weboldalon megjelenő elemeket. Az egyes egységeket érdemes div-ekben tagolni.

Legfontosabb tag-ek:

- <h1> ... <h6> : címsor. A h1 jelenti a legnagyobbat a h6 a legkisebbet. Minden címsor elé és mögé automatikusan bekerül egy üres sor.
- <a>... : a közé írt szöveg link-ként fog megjelenni.
- <p> : új bekezdést jelent. Minden bekezdés elé és mögé bekerül egy üres sor.
-
 : sortörés.
- <!--… --> : A közé írt szöveget a kód nem veszi figyelembe, megjegyzéseket tudunk így elhelyezni.

További fontos tulajdonsága a HTML nyelvnek, hogy nem veszi figyelembe a szóközöket [Huszár, 2009].

SVG (*Scalable Vector Graphics)*

Az SVG grafikus leírónyelvet a World Wide Web Consortium (W3C) egyik munkacsoportja 1998-ban kezdte el kidolgozni. A cél egy olyan szabvány kifejlesztése volt, amely XML[4]-t alkalmazva képes az ábrák leírására. Egy SVG dokumentum geometrikus alakzatokból álló ábrát tartalmaz szöveges formában. Ezek a grafikák később megjeleníthetőek egy weboldalon vagy egy erre alkalmas program (pl. IrfanView) segítségével. A grafikában szereplő elemek általában görbék, síkidomok vagy szöveges objektumok; azonban lehetőség van arra is, hogy nem vektoros grafikát is beépítsünk a rajzba. Mivel szövegesen van leírva a kép tartalma, a keresőprogramok könnyebben tudnak információhoz jutni az ábrával kapcsolatban, mintha hagyományos, raszteres ábrázolást használnánk. A vektorgrafika másik nagy előnye a szabad átméretezhetőség a minőség romlása nélkül. Az SVG erre is rugalmas lehetőséget biztosít a felhasználó számára, ezáltal lehetővé téve azt, hogy bármilyen felbontású eszközön ugyanolyan minőségben lehessen megtekinteni az ábrákat (pl. monitor, tablet vagy telefon) [W3C, 2005].

Mivel egy SVG kép nem egy csak egy elemből áll, amiben csak képpontok találhatóak, az így kirajzolt pontok, vonalak, felületek interaktívak is lehetnek: ha rámutatunk az egérrel, vagy rákattintunk egy elemre, az végrehajthat JavaScript utasításokat is. Az SVG megjelenítése a böngészőkben többnyire mindenhol működik, kivéve az Internet Explorer 8-as vagy régebbi verzióit, itt egy plugin segítségével ez a probléma is kiküszöbölhető.

Az SVG további nagy előnye a többi képmegjelenítő technikához képest, hogy tartalma közvetlenül beleírható a HTML-be: a megfelelő kulcsszavak közé írható be a vektorgrafika kódja.

A nyelv jelen pillanatban W3C ajánlási státuszban van (W3C recommendation - REC), ami a World Wide Web Consortium (W3C) legelőkelőbb fejlesztési állapota. Ez azt jelenti, hogy a nyelv egy teljes elméleti és gyakorlati tesztelésen, illetve vizsgálaton ment keresztül a konzorcium által. Ennek eredményeként alapvető szabványként lett jóváhagyva, ami a

[4] Az XML (Extensible Markup Language, Kiterjeszthető Jelölő Nyelv) a W3C által ajánlott általános célú leíró nyelv, speciális célú leíró nyelvek létrehozására.

támogatottság folyamatos növekedését eredményezi és lehetőséget nyújt más szabványokkal (DOM, XSL) való együttműködésre [W3C, 2005].

JavaScript

A JavaScript eredetileg egy olyan bővítmény volt, melyet csak néhány böngésző preferált. A nyelvet először a Netscape Communications Corporation által készített Netscape böngésző 2.0-ás verziója támogatta (1995 vége). A JavaScript egy kliensoldali szkriptnyelv, amelyet maga a böngésző értelmez és futtat.

Fontos megemlíteni, hogy a JavaScript nyelv nem egyenlő a Java nyelvvel. Elméletileg volt köze a névadásnál a Java programozási nyelvhez, viszont ez főleg marketing okokból történhetett.

Manapság a JavaScript programnyelv a webprogramozás során elengedhetetlenné vált. Ha dinamikus weboldalakat szeretnénk fejleszteni, elkerülhetetlen, hogy megismerjük, és jól használjuk ezt a nyelvet. Ezt még jobban megerősíti az a tény, hogy az AJAX (asynchronous JavaScript and XML) kifejlesztésével újra központba került a JavaScript nyelv. Fontosságát az is mutatja, hogy sok böngésző lassúságát, vagy éppen gyors működését a JavaScript feldolgozó motor okozza. Szinte mindegyik weboldalon található valamilyen kisebb, nagyobb szkript. A böngészőnek az oldal megjelenítésekor ezeket a fel kell dolgoznia, amelynek sebessége kihat az oldal betöltődési idejére, gyorsaságára.

A JavaScript a fejlesztőknek sok beépített objektumot biztosít, amelyek mindegyikének számos változója és függvénye van. Az objektumok elérést biztosítanak a HTML dokumentumunk minden lényeges entitáshoz (Document Object Model – DOM). A felhasználói bevitelt támogató HTML tag-ekre a programból is hivatkozhatunk, azaz a tag-ekkel definiált gombokat, szövegbeviteli sorokat vagy ablakokat JavaScript objektumokként érhetjük el.

JavaScript kódok egy HTML dokumentum tetszőleges részére beilleszthetőek, de praktikus ha a <head> részbe tesszük őket, mivel így a JavaScriptet nem alkalmazó böngészők semmiképpen sem fogják a program szövegét a weboldalra beolvasni. A kódsorokat a <script> … </script> tag-ek közé rakjuk [Ember, 2010].

CSS

A CSS az angol Cascading Style Sheets kifejezés rövidítése, jelentése rangsorolt stíluslapok. A stíluslapot általában egy szöveges fájlban írjuk meg és .css formátumban mentjük el, de akár a HTML dokumentumba is beágyazható. Fajtáját tekintve egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését adja meg. Ezen kívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb.

Az első verzió előkészületei 1994-1996 közé tehető. Ennek előkészítését ketten végezték, Hakon Wium Lie és Bert Bos. Az ő vezetésükkel 1996 decemberében megjelent a CSS1 azaz az első hivatalos verzió. 1998-ban a második verzió is elkészült CSS2 néven. A CSS3, azaz a harmadik verzió fejlesztései pedig még a napokban is folynak. A CSS olyannyira felülkerekedett a HTML formázási lehetőségein hogy a HTML 4.0 szabványból már ki is kerültek ezek a formázást segítő címkék.

A CSS egyik előnye, hogy mind a weboldal szerkesztői, mind pedig egy weboldal olvasói tudják vele szerkeszteni a lapok színét, betűtípusait, elrendezését, és más megjelenéshez kapcsolódó elemeit. A kialakítása során az egyik legfontosabb szempont az volt, hogy elkülönítsék a dokumentumok struktúráját (amelyet HTML-ben, vagy egy hasonló leíró nyelvben lehet megadni) a dokumentum megjelenésétől (amelyet CSS-vel lehet megadni) [Huszár, 2009]

CSS fájl felépítése

- Karakterkódolás beállítása:

  A stíluslapfájl legelső sorában ajánlott a karakterkódolásra vonatkozó információt írni:

  ```
  @charset "utf-8";
  ```

- Kijelölők, tulajdonságok, értékek:

  Az első rész után az egész stíluslap nem más, mint kijelölők (selector) és meghatározásblokkok (declaration) felsorolása. A meghatározás két részből áll: tulajdonságból (property) és értékből (value).

```
h1     {
        color: red;
}
```

Ebben az egyszerű példában a *„h1”* címke a kijelölő a *„color:red;”* sor pedig a meghatározás, ezen belül a color a tulajdonság (property), a red pedig ennek a tulajdonságnak az értéke (value).

A kijelölő egy vezérlő szerkezetet jelöl ki a HTML dokumentumban, és erre a kijelölt címkére vonatkozik a formázása.

- stíluslap beágyazása:

  Az elkészült stíluslap bármennyi HTML oldalhoz használható egyidejűleg. Viszont ahhoz, hogy a stíluslap hatással legyen rá, össze kell kötni a HTML fájlt és a stíluslapot. Ezt háromféleképpen tehetjük meg:

  - 1. Külső stíluslapok:

    Ha egy HTML fájlra alkalmazni kívánunk egy stíluslapot, akkor a fejrészbe (<head>...</head>) kell írni a következőt, ha a két fájl azonos mappában van:

```
<head>
<link href="stiluslapneve.css" rel="stylesheet"
type="text/css">
</head>
```

    A *„rel”* és a *„type”* attribútumnak mindig ez kell, hogy legyen az értéke, ha stíluslapot ágyazunk be, a *„href”*-nek pedig a stíluslap elérési útvonalát kell tartalmaznia a HTML fájlhoz képest.

  - 2. Belső stíluslapok

    Olyan stílusok megadásakor érdemes használni ezt a módszert, aminél tudjuk, hogy csak egyetlen weboldalhoz fogjuk felhasználni (ugyanis, ha egy stíluslapot több weboldalhoz is felhasználunk, akkor ajánlott külön külső fájlban tárolni őket, és a fenti módszerrel társítani a weboldalhoz). A *<head>...</head>* részbe kell beírni a stíluslapot a *<style>...</style>* címkék közé.

```
<head>
<style type="text/css">
h1 {
    color: red;
    text-align: center;
}
</style>
</head>
```

- 3. Szövegközi stílusok

  A *„style”* attribútumot bármelyik HTML vezérlőszerkezeten belül használhatjuk, ha egy stílus csak az adott elemre vonatkozik.

```
<h1 style="color: green; text-align: left;">
```

- Kommentek
  - Megjegyzést vagy kommentet a /* és a */ jelek közé írhatunk. Egy megjegyzés több sort is átfoghat. A megjegyzéseket nem veszi figyelembe a böngésző, semmilyen hatással nincsenek a kinézetre.

5. Felhasznált fájl formátumok

JSON, GEOJSON

A JSON (JavaScript Object Notation) egy kisméretű, szöveges formátumú adatcsere szabvány. A JavaScript szkriptnyelvből alakult ki, egyszerű adatstruktúrák és asszociatív tömbök reprezentálására (a JSON-ban objektum a nevük). A JavaScripttel való kapcsolata ellenére nyelv-független, több nyelvhez is van értelmezője. A JSON-t legtöbbször egy szerver és egy kliens számítógép közti adatátvitelre használják (legtöbbször AJAX[5] technológiával) az XML egyik alternatívájaként. Általánosságban strukturált adatok tárolására, továbbítására szolgál.

A JSON alap adattípusai:

- Szám (double a JavaScriptben, általánosságban implementációfüggő)
- Karakterlánc (string: idézőjelek közt, Unicode karakterekből (alapból UTF-8 kódolásban) balra dőlő törtvonallal (backslash, \) escape-elve)
- Boolean (true (igaz) vagy false (hamis))
- Tömb (értékek sorrendhelyes felsorolása vesszővel elválasztva, szögletes zárójelek között; az értékeknek nem kell azonos típusúnak lennie benne)
- Objektum (kulcs-érték párok rendezetlen gyűjteménye, amelyben ':' karakter választja el a kulcsot és az értéket, a párok egymástól vesszőkkel vannak elválasztva, a lista kapcsos zárójelek között van; a kulcsok mindig string típusúak, és különbözniük kell egymástól)
- Null (nincs adat)

A JSON-t az XML kis helyigényű alternatívájaként hirdetik, mivel általánosságban sokkal kisebb az elkészült fájl mérete.

A földrajzi adatok leírására szolgáló JSON típusú formátum a GeoJSON. A GeoJSON leírás fastruktúrában tartalmazza geometriai adatokat, leírókat, jellemzőket, illetve a gyűjtemény jellemzőket.

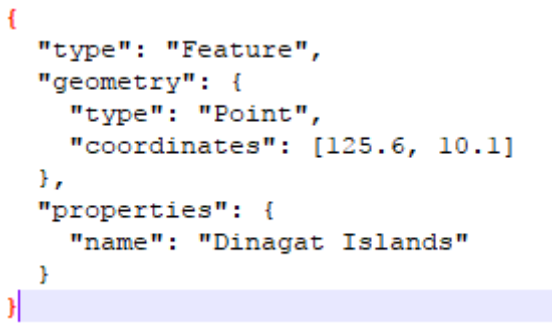
[5] Az Ajax (Asynchronous JavaScript and XML) lehetővé teszi, hogy a weblap teljes újratöltődés nélkül kis mennyiségű adatot cseréljen a szerverrel, és ennek hatására frissüljön. Az adat rögtön megjelenik a felhasználó számára, amint megérkezik a szerverről.

GeoJSON támogatja a következő geometriatípusokat:

- Pont
- Vonal
- Poligon
- MultiPont
- Multivonal
- MultiPoligon
- Geometriai gyűjtemény

A teljes GeoJSON adatok szerkezete megad egy, vagy több objektum geometriai adatait.

Lényege, hogy az alakzatok a töréspontok koordinátájával írja le, és ezek segítségével jeleníti meg őket.



4. ábra: geoJSON kód részlet

A formátum hátránya a mérete. A legkisebb alakzat leírása is nagy helyet foglal el, ezáltal nehézkessé válik a tárolása és kezelése. Ennek a kiküszöbölése miatt létrehozták ennek a formátumnak egy kibővített változatát, a TopoJSON-t.

A TopoJSON legnagyobb különbsége, hogy a leírt fájlok „össze vannak olvasztva” közös vonalszakaszok mentén (ív - arc), így ezzel kb. 85-90%-os méretcsökkenést tudunk elérni.

```
{
  "type": "Topology",
  "transform": {
    "scale": [0.036003600360036005, 0.017361589674592462],
    "translate": [-180, -89.99892578124998]
  },
  "objects": {
    "aruba": {
      "type": "Polygon",
      "arcs": [[0]],
      "id": 533
    }
  },
  "arcs": [
    [[3058, 5901], [0, -2], [-2, 1], [-1, 3], [-2, 3], [0, 3], [1, 1], [1, -3], [2, -5], [1, -1]]
  ]
}
```


5. ábra topoJSON kód részlet

A d3.JS mind a két állomány formátumot képes kezelni, viszont a dolgozatom elkészítéséhez geoJSON állományokat használtam. Ennek oka abban rejlik, hogy a térinformatikai szoftverek egyelőre csak ezt a formátumot támogatják [IETF, 2016].

ESRI shape fájl

Ezt az adattárolási formátumot az ESRI[6] cég fejlesztette ki 1990-ben. A tárolási mód leírását a cég nyilvánosságra hozta, és ma a legtöbb térinformatikai szoftver támogatja. A shape fájl valójában legalább három állományt (.shp, .shx és .dbf) jelent, melyeknek elnevezése azonos, mivel a feldolgozó rendszerek csak ebből tudják megállapítani a kapcsolatot. Az elnevezés a korábbi MS DOS operációs rendszer szabályait követi, ami 3 karakteres kiterjesztést jelent. Ezen kívül további azonos nevű, de más kiterjesztésű állományok is tartozhatnak a shape fájlhoz, amelyek karakterkódolási, indexelési és vetületi információkat tartalmaznak.

A három kötelező állomány közül az .shp kiterjesztés tartalmazza az alakzatok geometriai leírását. A fájl egy fix hosszúságú (100 bájtos) fejlécet, és alakzatonként egy változó hosszúságú rekordot tartalmaz.

Az .shx kiterjesztés állomány egy fix hosszúságú fejlécet és alakzatonként egy fix hosszúságú rekordot tartalmaz. Ez utóbbi rekordok az alakzatok shape állományon belüli kezdetét és hosszát tárolják. Ennek segítségével lehet az adatok között gyorsan keresni.

Végül a .dbf kiterjesztés állomány az alakzatokhoz tartozó egyéb szöveges, numerikus és dátum típusú információkat tárolja dBase formátumban.

[6] Environmental Systems Research Institute: https://www.esri.com/en-us/home

Ezeken kívül a szükséges fájlok mellett további fájlokból állhat az adathalmaz

- .gpj: grafikonok leírását tartalmazza
- .prj: a vetületi rendszer leírása WKT[7] formátumban
- .sbn .sbx: térbeli index formátum leíró
- .ain .aih: leíró adatok indexelése
- .ixs: geokódolási indexek
- .atx: leíró adatok indexelése (oszloponként)
- .shp .xml: Metaadatok XML formátumban
- .cpg: kódtábla és karakterkódolási információk

A shapefile-okban tárolt alakzatok lehetnek két-, három- vagy négydimenziósak, amelynél a negyedik dimenzió méretinformációkat tárol. Az alakzatok nem tartalmazhatnak körívet, és egy shapefile-ban csak egyfajta geometriai típus lehet: pont, vonallánc vagy sokszög. Ha többfajta alakzatot szeretnénk tárolni, akkor ezt több shapefile-ban lehetséges [ESRI, 1998].

[7] Well-Known-Text: objektumok vektoros geometriájának leírására szolgáló szöveges jelölőnyelv

6. Felhasznált programok

Notepad ++

A Notepad++ szabad forrású forráskód-szerkesztő program, ami több programnyelvet támogat. Szintaxis kiemeléssel, automatikus kiegészítéssel könnyíti a kódolást, a felülete testre szabható, makró felvételt és lejátszást is tud a számtalan más szövegszerkesztő tulajdonság mellett.

A program ingyenesen letölthető és használható. A letöltést a http://notepad-plus-plus.org oldalon kezdeményezhetjük.

Dolgozatom elkészítésénél a Notepad++ 7.5.2 verzióját használtam.

QGIS (Quantum GIS)

A QGIS az egyik vezető nyílt forráskódú térinformatikai szoftver, melynek fejlesztése még a 2000-es évek elején kezdődött el. A jelenlegi legújabb verzió a 2.18.12-es, melyet szabadon letölthetünk a projekt hivatalos oldaláról (www.qgis.org). A QGIS egyik nagy előnye, hogy fejlesztői szem előtt tartották a felhasználók igényeit és kívánságait a fejlesztés során. Éppen ezért a szoftver futtatható az összes jelentősebb operációs rendszeren: Windows, Mac OS-X, UNIX, Linux és már Androidon is.

Dolgozatom elkészítésénél a QGIS 2.18.12 (Las Palmas) verzióját használtam.

A program számtalan lehetőséget biztosít a felhasználónak, rengeteg problémát meg lehet oldani a segítségével. Mivel egy nyílt forráskódú programról beszélünk, így a felhasználóknak is lehetősége van különböző modulok elkészítésére, illetve ezek megosztására a többi felhasználó között. Ezek a modulok három csoportba oszthatók:

- Elkészült modulok: Teljesen elkészült, folyamatosan karbantartott modulok. Időnként frissítések is elérhetőek, illetve hiba esetén lehetőségünk van azt bejelenteni. Megkülönböztetünk kiemelten megbízható modulokat, ezek nevei zöld betűkkel vannak írva.
- Elavult modulok: Teljesen elkészült modulok, de ezekhez már nem érkeznek frissítések. A QGIS nem javasolja ezeknek a moduloknak a telepítését.

- Kísérleti modulok: A modul fejlesztése még folyamatban van, így számos hiba jellemezheti őket. A QGIS csak tesztelési szándék esetén javasolja telepíteni.

7. d3 Map Renderer modul

A d3 Map Renderer egy Python programozási nyelven készült QGIS modul, amely shape fájlokat tud átalakítani geoJSON formátummá úgy, hogy az beágyazható legyen különböző weboldalakba. Ezt úgy éri el, hogy a geoJSON fájlokhoz hozzárendeli a d3 JavaScript függvénykönyvtárát, illetve egy előre megadott HTML mintát kitölt az általunk megadott adatokkal. A modul a működése során a geometrián kívül a QGIS-ben létrehozott stílust is eltárolja. Segítségével különböző típusú tematikus térképeket tudunk létrehozni, amely alapját a d3 függvénykönyvtár szolgáltatja. A modul jelenleg kísérleti fázisban van.

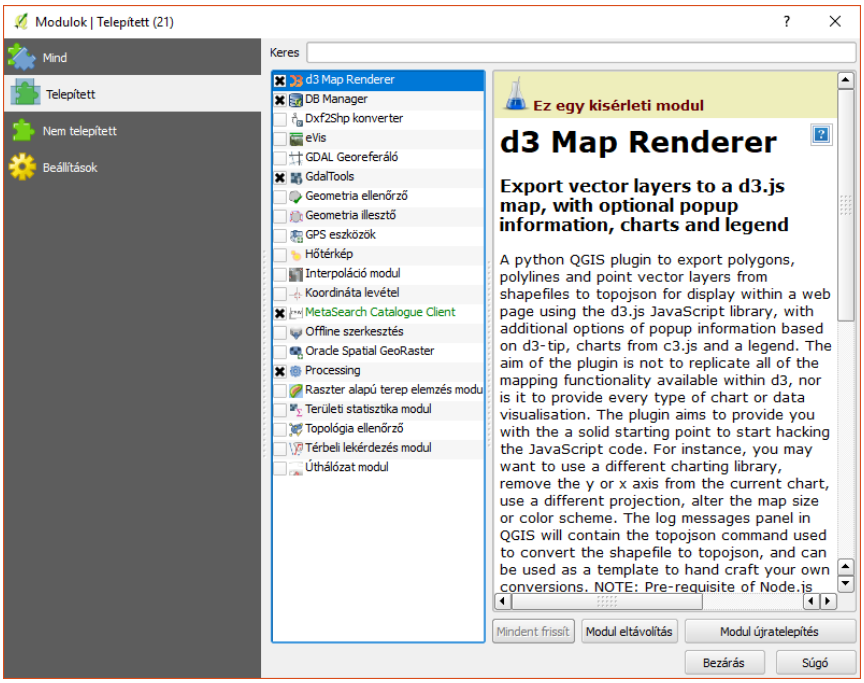
A modul leírásában a szerző (Simon Benten) felhívja a figyelmet arra, hogy a d3 Map Renderer nem használja ki a d3 függvénykönyvtár adta összes lehetőséget, csak egyfajta „kezdőlökést" ad azoknak, akik szeretnének ezzel foglalkozni. A legenerált állományt minden megkötés nélkül lehet a későbbiekben alakítani.

Használt verziószám: 0.10.3

d3 Map Renderer telepítése

Két féle módon van lehetőségünk a már feltelepített QGIS-hez hozzáadni a kívánt modult. A leggyorsabb mód, ha a QGIS-en belül kiválasztjuk a *'modulok'* fület, azon belül pedig a *'modulok kezelése és telepítése'* lehetőséget. A felugró ablakban rákeresünk a telepíteni kívánt modul nevére és a letöltés után automatikusan települni fog. Mivel jelen esetben kísérleti modulról van szó, ezért a beállítások fülnél be kell kapcsolni, hogy a kísérleti modulok is megjelenjenek a listában. Másik lehetőség a telepítésre, hogy ha közvetlenül a plugin weboldaláról[8] letöltjük, majd a QGIS plugineket tartalmazó mappájába másoljuk az állományokat.

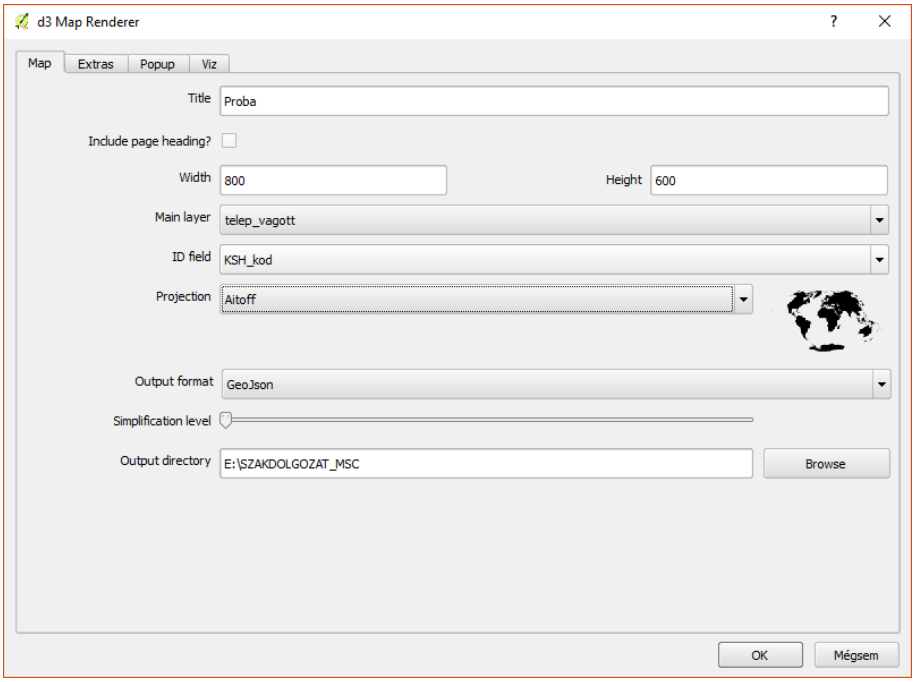
[8] http://maprenderer.org/d3/



6. ábra: A d3 Map Renderer modul leírása

A modul felépítése

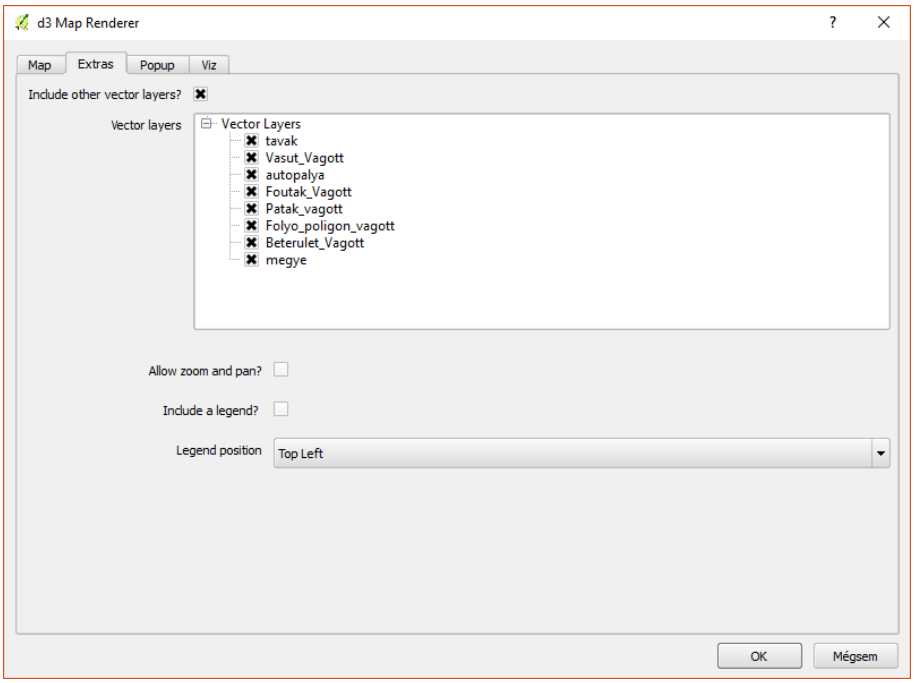
Miután telepítettük, az eszköztáron megjelenik egy D3 feliratú gomb, mellyel eltudjuk indítani a modult. A nyitóoldalon láthatjuk, hogy összesen négy fül érhető el, ezek sorrendben: Map, Extras, Popup és Viz.



7. ábra: A modul nyitóoldala

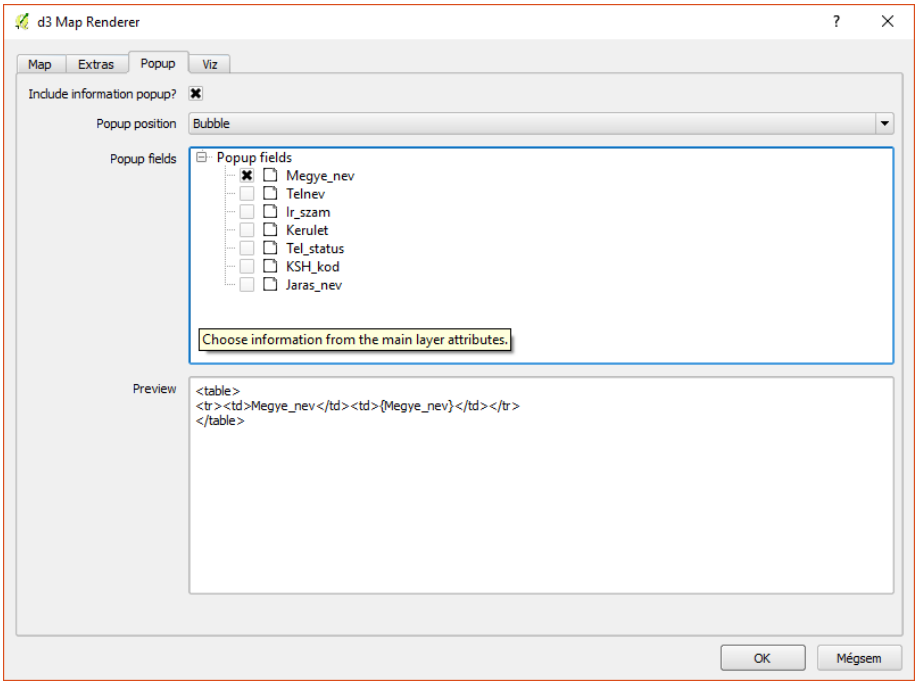
A Map (főoldalon) találhatóak azok a beállítások, amelyeket mindenesetben kötelező megadni:

- Title (Térkép címe): Az itt megadott cím lesz a weboldal címe.
- Width/Height (Térkép mérete): A térképlap szélességét és magasságát kell megadni, pixelben.
- Main Layer (Fő réteg): Azt a shape fájlt kell kiválasztani, aminek az adataival szeretnénk dolgozni. Az itt kiválasztott fájl attribútum adatait tudjuk később a térképen megjeleníteni.
- ID field (Elsődleges kulcs): Ki kell választani az attribútum táblázatból egy oszlopot, amely egyedileg azonosítja a táblában tárolt rekordokat. Itt csak olyan oszlopot válasszunk ki, amelyben nem szerepel ékezetes betű.
- Projection (Vetület): A használni kívánt vetületet kell megadni.
- Output format (Kimeneti fájlformátum): Egyelőre még csak a GEOJSON választható, későbbiekben a topoJSON opcióval kívánják bővíteni.
- Output directory (Mentés helye): a 'Browse' gomb megnyomásával megkell adni az elkészült fájl helyét.



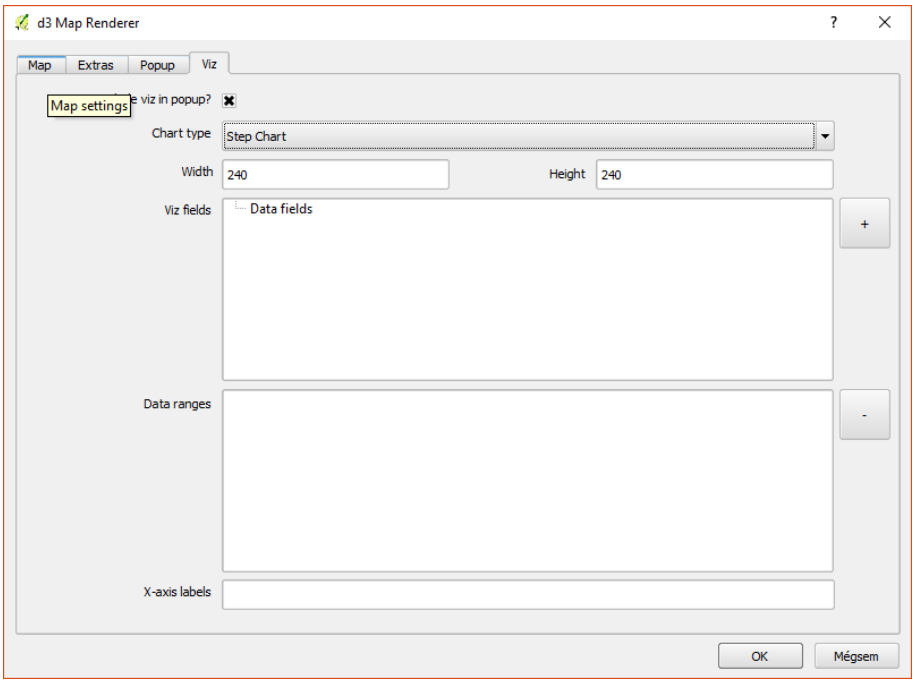
8. ábra: A modul Extras oldala

Az Extras oldalon további rétegeket adhatunk hozzá a térképünkhöz, illetve bekapcsolhatjuk, a *'zoom'* és a *'pan'* képességet. Az előbbi a tetszőleges nagyítást/kicsinyítést az utóbbi pedig a térkép szabad mozgatását jelenti. A *'legend position'* legördülő sávnál a jelmagyarázat elhelyezését lehet beállítani.



9. ábra: A modul Pop-up oldala

A Pop-up oldalon lehetőségünk van létrehozni a térképen kattintásra felugró ablakokat (Pop-up ablakokat). A Main layer-nek megadott shape fájl attribútumai közül van lehetőségünk választani. A *'popup position'* legördülő listánál megadhatjuk, hogy milyen típusú legyen a megjelenő ablak. Kettő lehetőségünk van: *„bubble"*–ekkor egy buborék ugrik fel az objektum fölé; vagy lehet *'external'*, ekkor pedig a térképlap bal alsó sarkában jelennek meg a kiválasztott adatok.

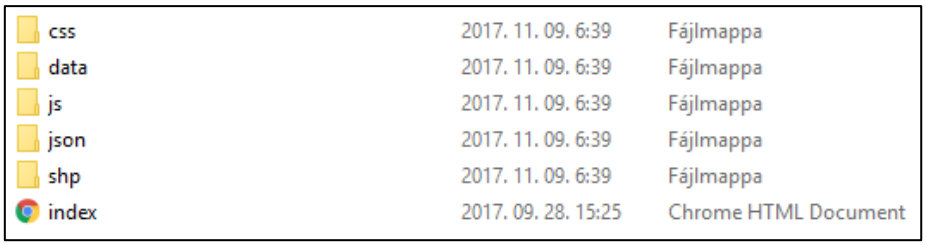


10. ábra: A modul Víz oldala

A Viz (visualization) oldalon különböző típusú diagramokat hozhatunk létre a fő réteg attribútumaiból. A *'Chart type'*-nál a 16 db előre megírt diagram típusból választhatjuk ki az általunk használni kívántat. A *'Width'* és *'Height'*-nál a diagram méretét tudjuk megadni, pixelben. A *'Viz fields'* ablakban láthatunk minden olyan adatsort a fő réteg attribútum táblázatából, amely numerikus adatokat tartalmaz. Miután kiválasztottuk a használni kívánt adatsorokat, meg kell adnunk a diagramunk nevét, majd a *'Data ranges'* ablakban láthatunk egy összesítést a grafikonunkról. *'X-axis labels'*-nél megadhatjuk az x tengely megírásait. A diagramok létrehozását részletesebben a *Magyarország működő bányáinak száma 1970 és 2005 között* című térképnél mutatom be [Benten, 2015].

Létrehozott fájlok

Ha minden kívánt beállítást elvégeztünk, az 'OK' gombra kattintva a modul legenerálja a térképünket. Az így létrejött mappák és fájlok tartalmát és felépítését külön-külön mutatom be. A modulban minden elvégzett beállítás a létrehozott fájlokban megváltoztatható, illetve kibővíthető.



11. ábra: A modul által létrehozott állományok

css (mappa)

Ebben a mappában található az összes .css kiterjesztésű fájl. Összesen öt darab lesz a mappában, mindegyikben valamilyen térképi elem stílusát tudjuk módosítani. A következő felsorolásban bemutatom, hogy melyik fájl minek a tulajdonságát tartalmazza.

- c3.min.css: A térképen megjelenő grafikonok tulajdonságai
- color.css: A térképen megjelenő objektumok tulajdonságai.(Kontúr vonal vastagság és szín, kitöltés színe, átlátszóság stb.)
- legend.css: A jelmagyarázat tulajdonságai
- map.css: A térképi keret tulajdonságai
- tip.css: A megjelenő pop-up ablakok tulajdonságai

A modul minden esetben legenerál minden .css fájlt, viszont ha például nem hoztunk létre jelmagyarázatot, akkor a legend.css fájl üres marad.

data (mappa)

Ebben a mappában találhatóak a térképen megjelenített adatokat tartalmazó .csv kiterjesztésű fájlok. Amennyiben nem állítottunk be jelmagyarázat és/vagy pop-up ablakot, akkor ez az állomány üres marad. A mappa felépítése:

- info.csv: Ebben a fájlban található az összes attribútum melyet a diagramok és a pop-up ablakok felhasználnak
- legend.csv: Ebben a fájlban található a jelmagyarázat elemeinek a leírása

Az legend.csv fájl felépítése:

```
Width,Height,Color,Text

,,,Népsűrűség
,,,
20,20,l0r0, 56-74
```

```
20,20,l0r1, 75-84
20,20,l0r2, 85-94
20,20,l0r3, 95-104
20,20,l0r4, 104-
20,20,l1r0, belterület
```

Az adatokat vesszővel elválasztva kell megadni. Négy adat feltüntetésére van lehetőségünk a jelmagyarázaton belül:

1) A jelet befoglaló négyzet szélessége és magassága
2) A jelhez tartozó szín CSS kódja (amelyet a color.css-ben kell definiálni)
3) A jelhez tartozó magyarázó szöveg tartalma

Ha egy adatot kihagyunk, akkor a megjelenítés során is ki fog maradni az adott rész, ezáltal létrehozhatunk csak szövegekből álló sorokat is, így adva címet az adott csoportba tartozó jeleknek.

js (mappa)

Itt található az összes JavaScript állomány:

- c3.min.js: Különböző fajta diagramok generálását leíró állomány.
- map.legend.js: A jelmagyarázat leírását szolgáló JavaScript állomány
- map.tip.js: Ez a fájl írja le azt, hogy ha rákattintunk egy térképi elemre, akkor valamilyen esemény következzen be, például jelenjen meg a „buborék” ablak.

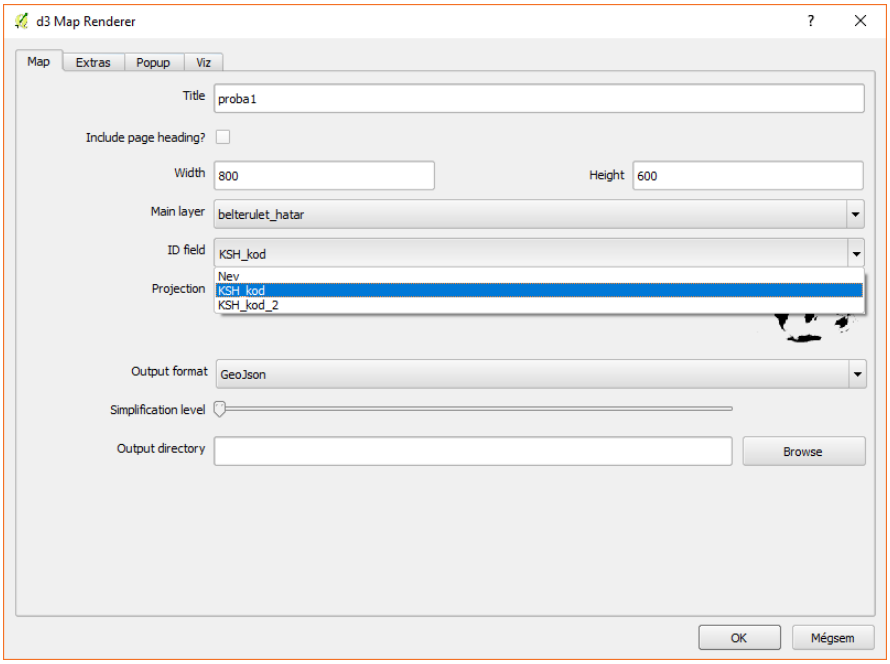
shp (mappa)

A d3 Map Renderer kiindulási alapja – mint többnyire a QGIS moduloknak – a shape fájl(-ok). A modul az shp mappában a fent említett file-ok „butított” változatát tárolja. A „butított” jelző jelen esetben azt jelenti, hogy csak azokat az attribútumokat hagyja meg a kiindulási fájlból, amelyek valójában használunk, a többit vagy törli az állományból vagy egy .csv fájlban tárolja őket.

	Nev	KSH_kod	KSH_kod_2
1	Felsőszölnök	2328	23287
2	Alsószölnök	2254	22549
3	Kétvölgy	1934	19345
4	Szakonyfalu	2093	20932

12. ábra: Az eredeti belterület_hatar shape fájl attribútum táblázata (részlet)

A fenti táblázat-részletben látjuk, hogy három oszlopunk van, amelyből a „Nev” mező adatait jelenítenénk meg a településre való kattintás után a weboldalon. A modul használata során beállítjuk, hogy melyik rétegről szeretnénk megjeleníteni adatokat az előreugró ablakban. Ehhez egy olyan egyedi azonosítót (ID Field) kell választanunk, amely nem tartalmaz ékezetes betűket.



13. ábra: A modulban a főréteg és az egyedi azonosító kiválasztása

Jelen példánkban ez a KSH_kod oszlopot jelenti. A kiválasztott mezőt az új shape fájl eltárolja, míg azt a mezőt, amiből az adatokat szeretnénk kiíratni (Nev), áthelyezi az info.csv fájlba az elsődleges kulcsokkal párosítva.

	A
1	Nev,KSH_kod
2	Felsőszölnök,2328
3	Alsószölnök,2254
4	Kétvölgy,1934
5	Szakonyfalu,2093

14. ábra: info.csv fájl (részlet)

A felesleges oszlopok törlésén kívül létrehoz egy új „d3Css”mezőt. Ez az adatsor egységes kóddal rendelkezik. Ez a kód egy színkódnak felel meg, amelyet a belterület kitöltéséhez használt színhez társít a color.css fájlban.

	KSH_kod	d3Css
1	2328	l1r0
2	2254	l1r0
3	1934	l1r0
4	2093	l1r0

15. ábra: A modul által létrehozott shape file attribútum táblázata (részlet)

Ezen kívül eltérés van még a shape file-hoz tartozó .prj fájlban is, itt már értelem szerűen az új vetület leírása jelenik meg.

json (mappa)

Ebben a mappában találhatóak a felhasznált shape fájlok geoJSON formátummá alakított változatai.

index.html

Az állomány fő tartalma. Általános felépítése, sorrendben:

- <head>...</head> rész: Itt definiáljuk a weboldal címét, illetve itt hívjuk meg a különböző .css fájlokat és JavaScript függvénykönyvtárakat. A d3.JS függvénykönyvtárat csak meghívjuk, de nem töltjük le, ezért internet kapcsolat szükséges a megjelenítéshez a weboldal működéséhez.
- A <body> legfelső részében helyezzük el a weboldalon megjelenített szövegeket. Itt van lehetőségünk a weboldalt formázni.

- <table>…</table>: Itt tudjuk megadni a felugró pop-up ablakok által megjelenített adatokat. Ha például a „Település neve: Tata” feliratot szeretnénk megjeleníteni, akkor begépeljük a szöveg elejét, majd egy hivatkozást adunk meg neki, mely az info.csv fájl *„Nev”* oszlopára mutat.

```
<tr><td>Település neve:</td><td>{Nev}</td></tr>
```

- Az index.html utolsó és egyben leghosszabb szakasza a <script>…</script> között elhelyezkedő tartalom. A modul úgy generálja le a fájlt, hogy a könnyebb értelmezés miatt különböző címekkel látja el az azonos logikai csoportba tartozó kódokat. A következő felsorolásban a legfontosabb címekhez tartozó szakaszokat mutatom be:
    - *Basic settings:* A térkép ablak méreteit lehet megadni, módosítani.

```
var width = 1200;
var height = 800;
var _Data = void 0;
```

    - *Projection:* A térképi vetületet lehet itt definiálni, illetve kirajzoltatni.

```
var projection = d3.geo.mercator()
      .center([19, 47])
      .scale(5500)
      .translate([width / 2, height / 2]);
      var path = d3.geo.path().projection(projection);
      var graticule = d3.geo.graticule()
      .step([1, 1]);
```

    - *Map container:* A meglévő képekből SVG állományt hoz létre, illetve meghívja a zoom szkriptet.

```
var svg = d3.select("#map")
      .append("svg")
      .attr("id", "mapSvg")
      .attr("width", width)
      .attr("height", height)
      .on("click", hideTip)
      .append("g");
svg.call(d3.behavior.zoom()
      .scaleExtent([1, 40])
      .on("zoom", onZoom));
```

    - *Vector Group:* A nagyítási („zoom”) funkció hiba nélküli futása miatt, minden grafikát egy *„g”* jelű csoportba rakjuk.

```
var vectors = svg.append("g")
.attr("class", "polygon");
```

- *Vector polygons:* Különböző sorszámú üres csoportokat hozunk létre, melyeket a *„g”* jelű csoportba rakunk. Minél több rétegünk van, annál több csoportot kell létrehoznunk.

```
var vectors0 = vectors.append("g");
var vector0 = void 0;
```

- *Download:* Itt olvassuk be a .json és a .csv fájlokat

```
queue(2)
	.defer(d3.json, "json/megye.json")
```

- *Add all vector features:* A Vector poligons szakaszban létrehozott üres csoportokat feltöltjük a *Download* szakaszban beolvasott geoJSON fájlokkal, illetve minden egyéb, a különböző rétegekhez tartozó módosítást itt tudunk elvégezni. (pl.: különböző események generálása)

```
vector0 =
vectors0.selectAll("path").data(object0.features);
vector0.enter()
	.append("path")
	.attr("id", function (d) { return
	d.properties.KSH_kod_2; })
	.attr("d", path)
	.attr("class", function (d) { return
	d.properties.d3Css; });
```

- *Show a tool tip for the selected element*: Itt adjuk meg azt, hogy a Pop-up ablak honnan vegye ki az adatokat:

```
function showTip(id) {
	var obj = _.findWhere(_data, {KSH_kod: id.toString()});
	tip.html(info(obj))
	.show();}
```

- *Zoom/pan:* Itt tudjuk megadni a nagyítás (’zoom’) utáni tulajdonságokat, csoportonként:

```
vector0.style("stroke-width", 0.26 / d3.event.scale);
```

A dolgozatom következő fejezeteiben a d3 Map Renderer modul segítségével elkészített és javított, különböző típusú tematikus térképeket fogom bemutatni. Minden térképnél csak a készítés azon fázisát emelem ki, amely eltér a többitől.

8. Térképkészítés d3 Map Renderer modullal

Magyarország úthálózata

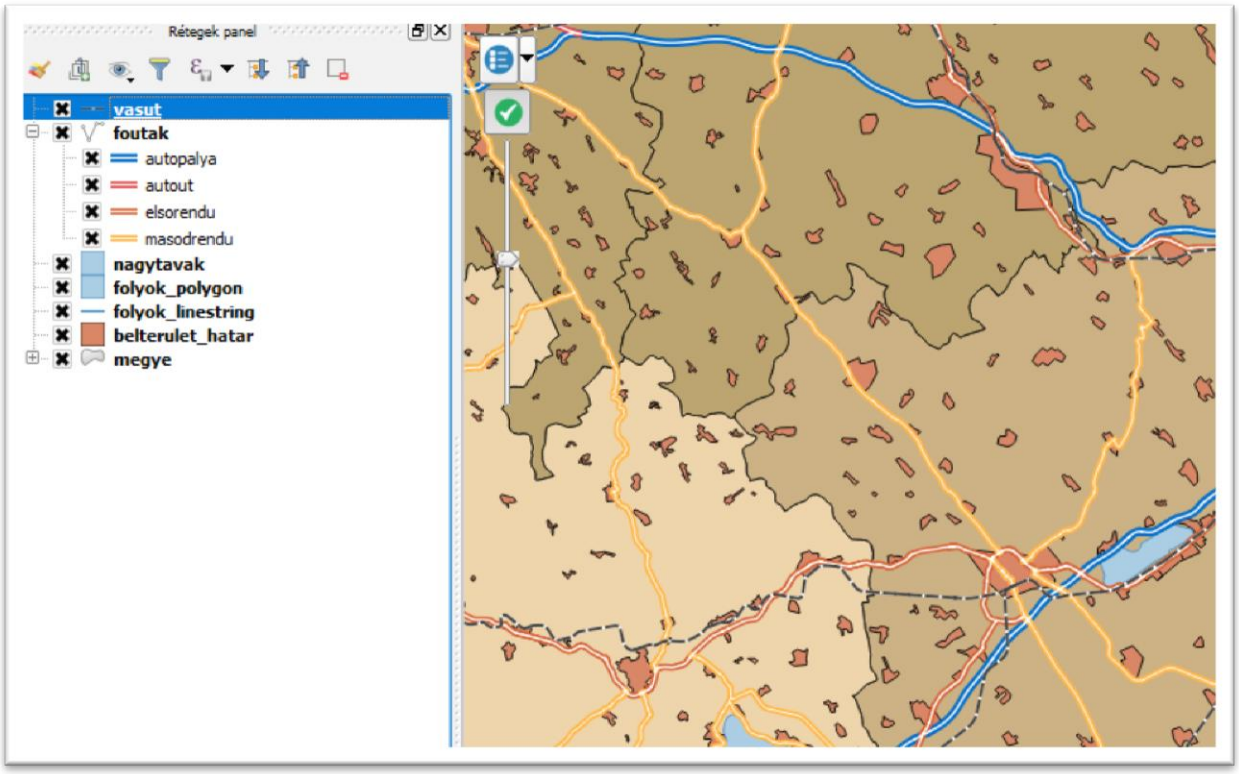
Az első térkép, amit a dolgozatomhoz készítettem egy teljes Magyarországot ábrázoló tematikus térkép. A térkép fő célja a népsűrűség megye szintű bemutatása, az út és a vasút hálózat kategorizált megjelenítése, illetve a település poligonokra való kattintással a település nevek kiíratása. Első lépésben létrehoztam QGIS-ben a térképet, majd legeneráltam a d3 Map Renderer modullal, végül pedig a létrehozott fájlokat kézzel módosítottam.

Felhasznált állományok: megye.shp, belterulet.shp, vasut.shp, tavak.shp, folyok.shp

A térkép elkészítésének folyamatai közül részletesen csak a vonalas objektumok létrehozását mutatom be.

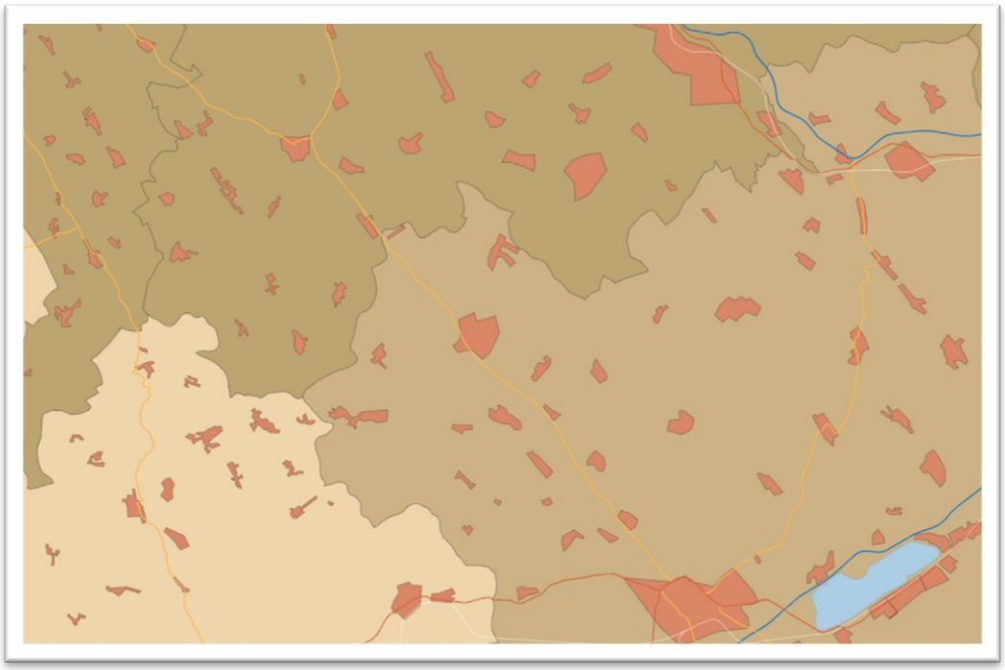
Térképeken a vonalas objektumok megjelenítéséhez számos szabályt kell figyelembe vennie a térkép készítőjének. Az egyik legfontosabb ilyen szabály, hogyha kategorizálva szeretnénk felvenni a térképre az objektumokat, akkor a térképről első ránézésre le lehessen olvasni a kategóriák közti különbséget, illetve a vonalas objektumok közötti hierarchiát. Az ilyen fajta hierarchia kifejezésére utak esetében a kétvonalas megjelenítés a legelfogadottabb megoldás.

A Magyarország úthálózata térképnél az utakat négy kategóriába soroltam. Autópálya, autóút, elsőrendű illetve másodrendű főút. Mind a négy kategóriát kétvonalas objektumként raktam fel a térképre. A legvastagabb vonal 1,5 mm széles, majd kategóriánként 0,2 mm-rel vékonyodnak. Kék, piros, narancs és sárga színeket használtam, melyeket egy fehér vékonyabb vonal egészít ki. Vasutaknál csak az elsőrendű vasútvonalakat jelenítettem meg, amelyet egy fekete teli alapvonallal és egy fehér szaggatott vonallal ábrázoltam.



16. ábra: Vonalas objektumok megjelenítése QGIS-ben

Általános hiányossága a d3 Map Renderer modulnak, hogy a kétvonalas objektumoknál csak az alsó, vonalat jeleníti meg, míg a felső réteget törli. Mivel a térkép értelmezhetősége így nagymértékben romlik, ezért erre a hibára megoldást kellett találni. Kisebb hiba, hogy a meglévő vonalvastagságot jelentősen csökkenti. (A 3.0 pontos vonalvastagságot 0.65 ponttá alakítja.)



17. ábra: A modul által generált úthálózat

A probléma megoldásának menete: A modul által létrehozott vonalas objektumok koordinátáit egy új „rétegre” kell másolni, majd új stílust kell létrehozni az új elemeknek.

Ezeket a műveleteket a vasút példáján mutatom be.

Az index.html fájlban létrehoztam egy új üres csoportot, amelyet *„vector 8”*-nak neveztem el. A HTML fájlból a csoportokat sorrendben olvassa ki a program, tehát a legalsó csoport lesz a legfelső réteg a térképen, így a felső fehér szaggatott vonalat hoztam létre.

```
var vectors8 = vectors.append("g"); //vasút felső
var vector8 = void 0; //vasút felső
```

Az állományhoz tartozó color.css fájlban létrehoztam egy új stílust *„vasút_felso”* néven, mely a fehér szaggatott, 2,5 mm széles vonalat definiálja.

```
.vasut_felso  { pointer-events: none; stroke: #ffffff; stroke-width:
2.5; stroke-opacity: 1.0; stroke-dasharray: 5; fill-opacity: 0.0; }
```

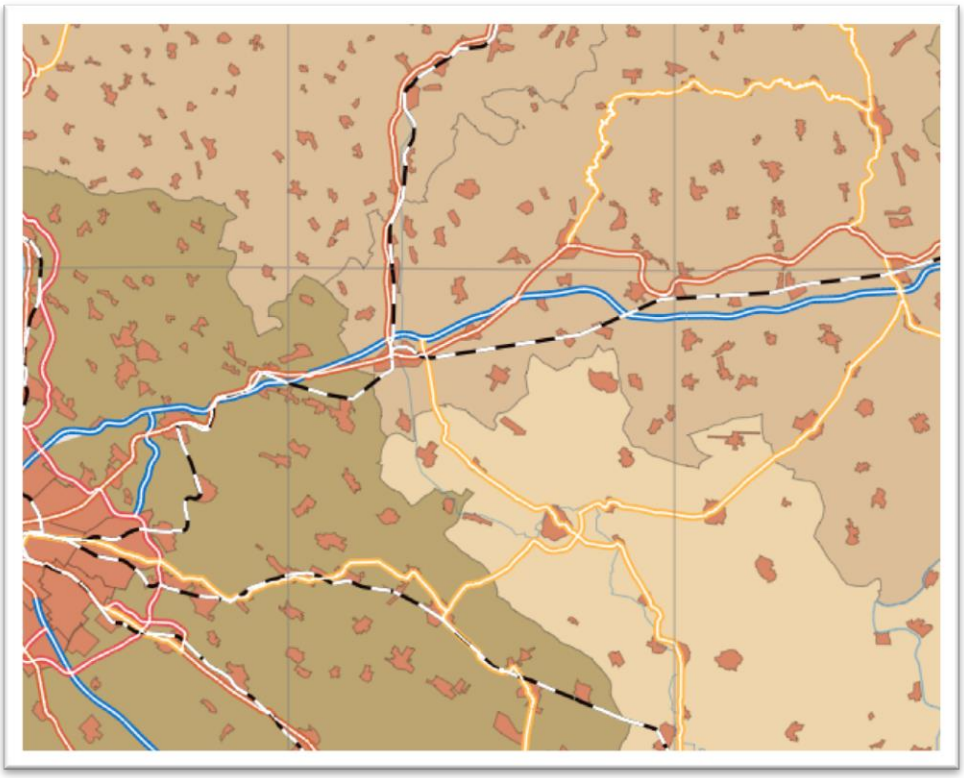
Az üres, *„vector8”* nevű csoportot feltöltöttem a már meglévő vasút csoport geometriai adataival, ezáltal lemásoltam a már meglévő teli fekete vonalat. A különbség az, hogy a kód azon részénél, ahol rá mutat a használni kívánt stílusra, az imént létrehozott *„vasut_felso”* stílust adtam meg neki.

```
vector8 = vectors8.selectAll("path").data(object6.features);
```

```
vector8.enter()
.append("path")
.attr("id", function (d) { return d.properties.KSH_kod_2; })
.attr("d", path)
.attr("class", "vasut_also");
```

A két vonalnak a színén és vastagságán felül be kellett állítani a *„zoomolás”* utáni vonal vastagságokat is. Ezt az index.html „zoom/pan” részében lehet megtenni. Itt is hozzá kellett adni az újonnan létre hozott csoportunkat és be kellett állítani a vonalak vastagságát.

```
vector8.style("stroke-width", 1.2625 / d3.event.scale);
```



18. ábra: Javított úthálózat

Az így létrejött térkép általános hibája, hogy a vonalas objektumok alá- és fölérendeltségét nem lehet módosítani, így előfordulhatnak olyan hibák, miszerint az autópályákat elsőrendű- illetve másodrendű főutak kereszteznek. Ezt csak úgy lehet kiküszöbölni, ha QGIS-ben, az alaptérkép elkészítésekor minden azonos kategóriába tartozó vonalas objektumokat külön rétegre rakunk. Ez esetben az alá- és fölé-rendeltség a D3.JS függvénykönyvtár segítségével létrehozott térképeken is megmarad.

Jelmagyarázat létrehozása

A d3 Map Renderer modul jelmagyarázat generálására is képes. Hátránya, hogy a csak a „főrétegekhez”, azon belül pedig csak egy attribútumhoz képes ezt megtenni, ezért a jelmagyarázat jelentős részét manuálisan kell létrehoznunk.

A jelmagyarázat generálása során két fájl jön létre, egy legend.css és egy legend.csv fájl.

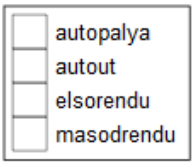
A legend.css fájlt a következőképpen néz ki:

```
.legendouter {
stroke: #000;
stroke-width: 0.8;
stroke-opacity: 1;
fill: #fff;
}

.legend rect, .legend ellipse {
stroke: #000;
stroke-width: 0.4;
stroke-opacity: 1;
}

.legend text {
fill: #000;
font-size: smaller;
}

.legend text.left { text-anchor: start; }
.legend text.right { text-anchor: end; }


data->legend.csv
```

Az legend.csv fájl felépítése:

```
Width,Height,Color,Text

,,,Népsűrűség
,,,
20,20,l0r0, 56-74
20,20,l0r1, 75-84
20,20,l0r2, 85-94
20,20,l0r3, 95-104
20,20,l0r4, 104-
20,20,l1r0, belterület
```

Problémát jelent a vonal típusú objektumok jelmagyarázatának az elkészítése, ugyanis ilyen fajta jelmagyarázatot a modul nem képes generálni.



19. ábra: A modul által generált jelmagyarázat

Ennek az oka a következő: A modul egy sémát használ a jelmagyarázat generáláshoz, mely nem veszi figyelembe az objektum alaptulajdonságait. Függetlenül attól, hogy felületi vagy vonalas objektumokhoz szeretnénk jelmagyarázatot generálni, ugyanazokkal a lépésekkel fogja ezt megtenni. Veszi az adott objektumok kitöltési színét és egy előre meghatározott méretű téglalapot kitölt azzal a színnel. A felületi módszernél ez a séma tökéletesen működik, mivel ott a poligonként felvett objektumokat kitöltéssel látjuk el, melyek segítségével tudjuk megkülönböztetni az adott típusú objektumokat. Ezzel szemben, ha vonaltípusú objektumokat veszünk fel a térképre, azokat – általában – nem poligonként ábrázoljuk, hanem vonalas objektumként, melyhez vastagságot és színt rendelünk hozzá. Ebben az esetben a modul nem tud kitöltést rendelni a jelmagyarázatban létrehozott téglalapoknak ezért üresen hagyja őket. A color.ccs-ben létrehozott stílus definíció:

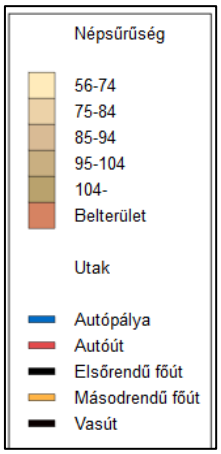
```
.l5c0 { pointer-events: none; stroke: #006bc6; stroke-width: 3.5;
stroke-opacity: 1.0; stroke-dasharray: ; fill-opacity: 1.0; }
```

Tökéletes megoldást erre a problémára sajnos nem találtam, megoldásához további programozás lenne szükséges a map.legend.js fájlban. A jelmagyarázatban lévő téglalap magasságának a méretét a legend.csv fájlban „5”-re módosítottam így az utak vonaltípusú jellegzetességét megtudtam jeleníteni.

```
,,,Utak
,,,
20,5,jel_kek, Autópálya
20,5,jel_piros, Autóút
20,5,jel_autonarancs, Elsőrendű főút
20,5,jel_autosarga, Másodrendű főút
20,5,jel_vasut, Vasút
```

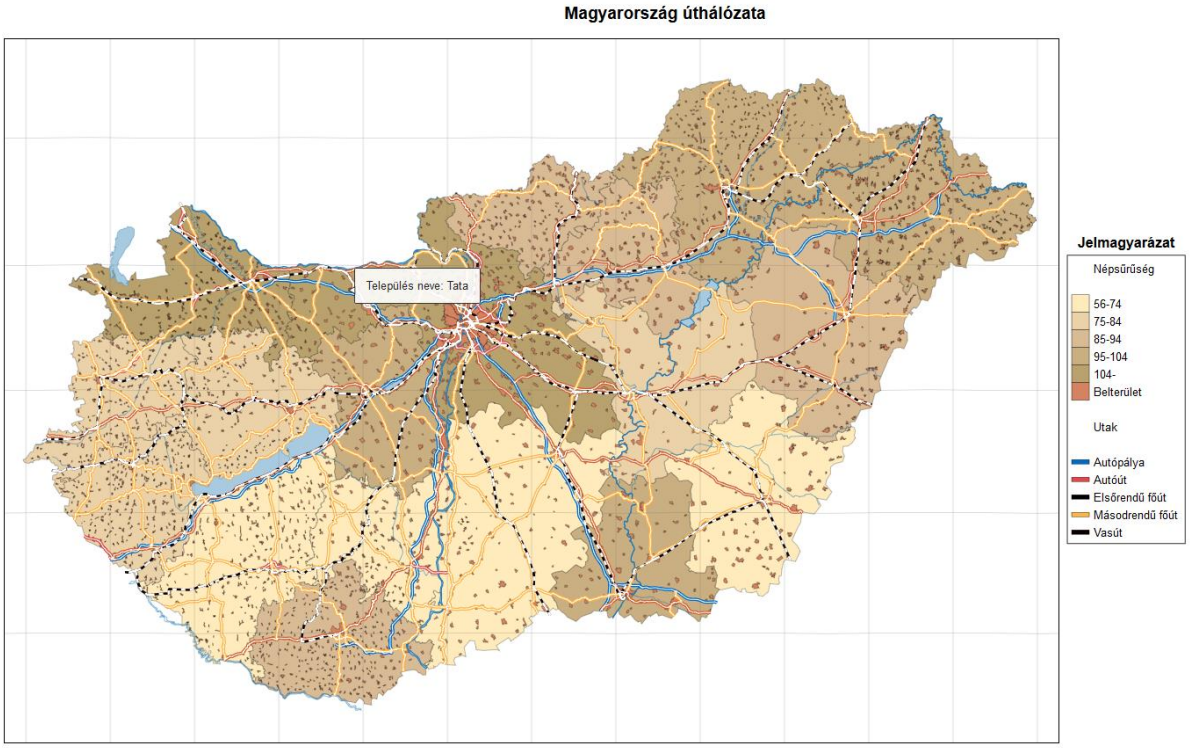
Kitöltésére a color.css fájlban új stílusokat hoztam létre, minden stílusnak az adott vonalas objektum alsó vonalának a színét adtam.

```
.jel_kek{ pointer-events: none; stroke: #006bc6; stroke-width: 0.26; stroke-
opacity: 1.0; stroke-dasharray: ; fill: #006bc6; fill-opacity: 1.0; }
```



20. ábra: Javított jelmagyarázat

Az létrehozott jelmagyarázat így a két vonallal ábrázolt jelekből csak az alsó, vastagabb vonalat jeleníti meg, így nem egyezik meg a térképen használt jelkulccsal. Esetleges megoldása lehet a problémának, ha a jelmagyarázathoz tartozó JavaScriptet úgy módosítjuk, hogy kettő egymástól független téglalap helyezkedjen el egymáson. Így lehetőségünk lenne az utakat hiba nélkül megjeleníteni.



21. ábra: Az elkészült Magyarország úthálózata című térkép

A fent említett hibákon kívül meg kell jegyezni, hogy ha az imént bemutatott térkép kartográfiailag helyes létrehozása rendkívül sok időt, és a JavaScript nyelv mély szintű

ismeretét igényli a térkép készítőjétől, ezért véleményem szerint ezzel az eszközzel célszerű kevésbé összetett térképek elkészítésére törekedni.

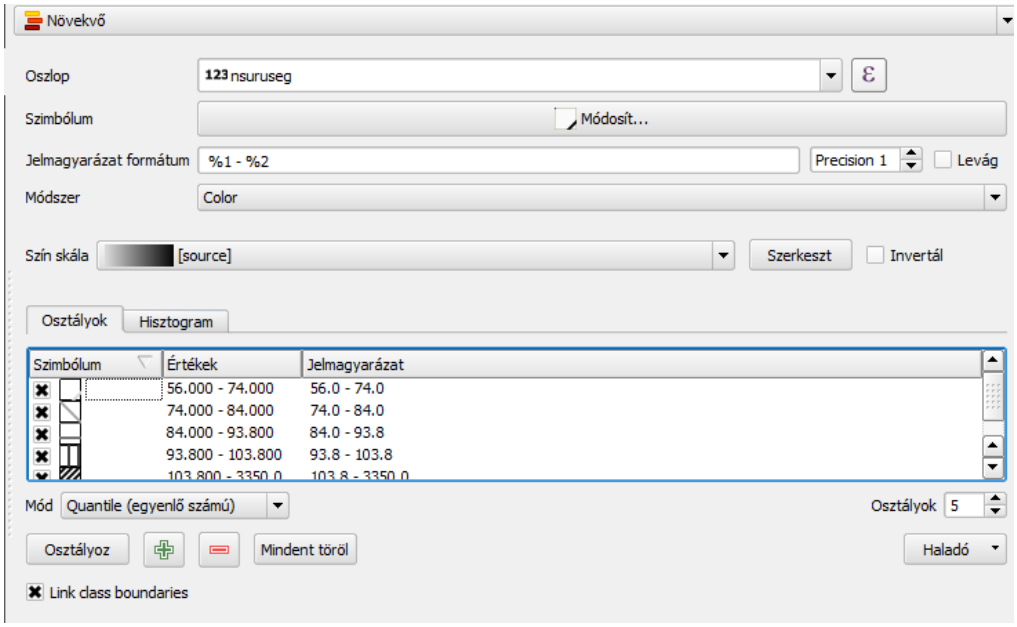
Magyarország népsűrűsége

A második térképemnél a kevésbé összetettség miatt, csak a felületkartogram módszerre koncentráltam. Az előző térképnél lehet látni, hogy a különböző értékek szerinti felületszínezés szépen megvalósítható, viszont a Magyarország népsűrűsége térképemnél ezt egy kicsit tovább gondoltam. A való életben, sok esetben a térképkészítő a nyomtatás költségeinek csökkentése miatt fekete-fehérben készíti a térképeket, ezért szerettem volna kipróbálni ugyanezt webes környezetben is: a szürke szín árnyalataiban készítettem egy népsűrűség térképet Magyarország megyéiről. Ebben az esetben a legjobb megjelenítési módszer, ha a különböző adatcsoportokat más-más felületi raszterrel különítjük el.

QGIS-ben egy ilyen térképet kis idő ráfordítással, viszonylag egyszerűen lehet elkészíteni.

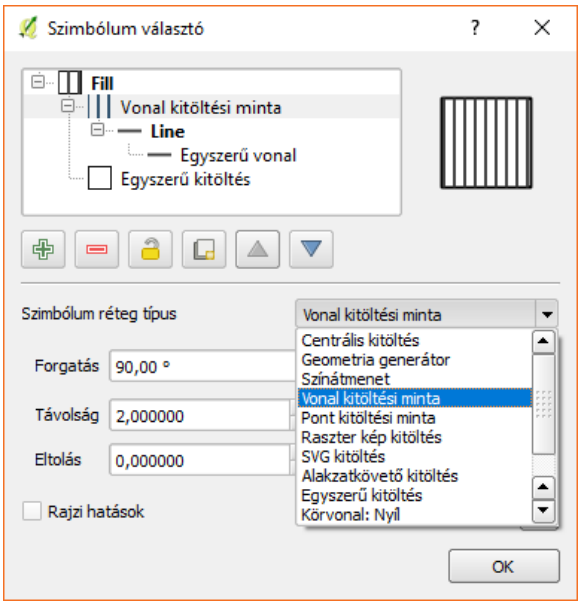
Felhasznált állomány: megye.shp

A megye.shp-hez tartozó attribútum táblázatot feltöltjük a népsűrűségi adatokkal, majd a réteghez tartozó stílust *„növekvő”*-re állítjuk és megadjuk, hogy az *„nsuruseg”* nevű oszlop alapján szeretnék kategorizálni a megyéket. Lehetőségünk van megadni a csoportosítási osztályokat, tetszőleges értékek szerint.



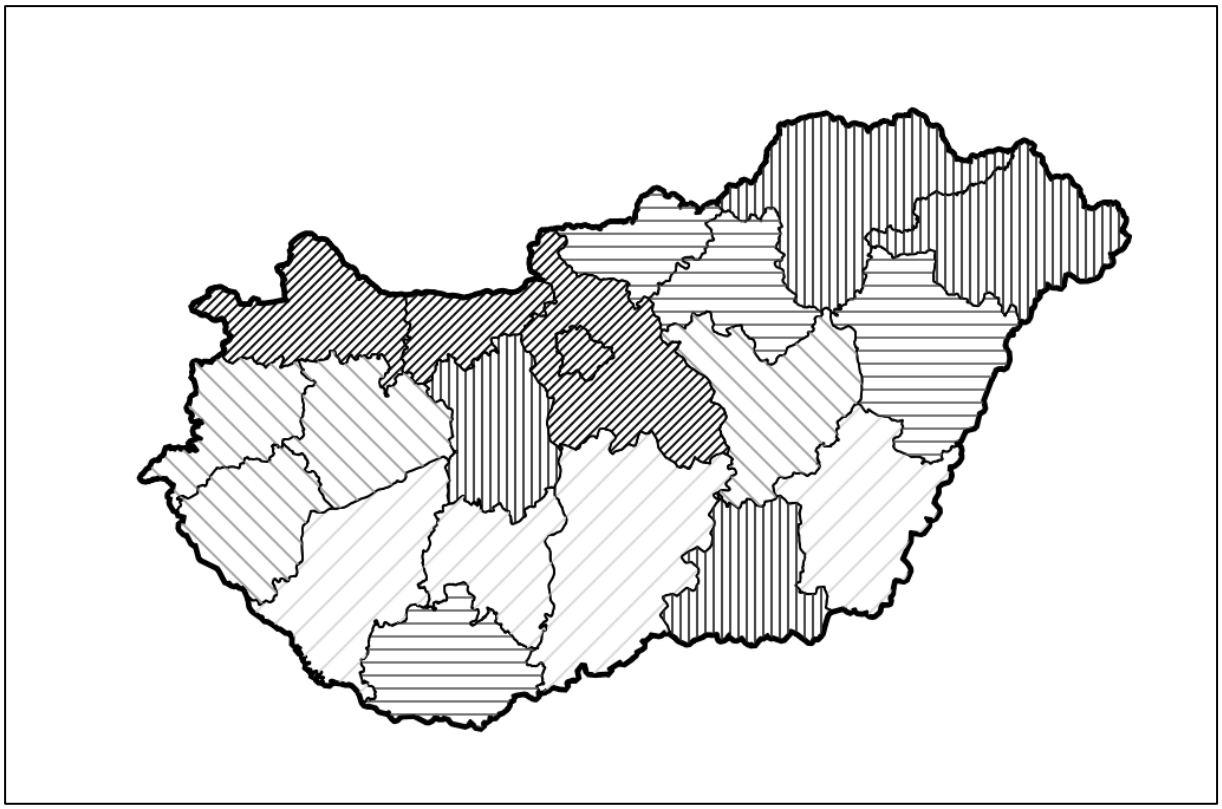
22. ábra: A megye.shp-hez tartozó stílus beállítások

Az egyes csoportok alapbeállítási stílusát – teli kitöltés – meg kell változtatni *„Vonal kitöltési minta”*-ra. Az így létrejött csíkozási vonalak színét, dőlését, vastagságát és egymástól való távolságát kedvünk szerint tudjuk állítani.



23. ábra: Felületi raszter tulajdonságainak beállítása

Az így elkészült térkép megfelel az előzetesen megadott szürkeárnyalatos kritériumnak és a térképolvasó számára egy könnyen és gyorsan értelmezhető térképet kaptunk.



24. ábra: QGIS-ben létrehozott felületi módszerrel ábrázolt, Magyarország népsűrűségét bemutató térkép

d3 JavaScript függvénykönyvtárral készített térkép

A d3 Map Renderer modul nem képes a sraffozással kitöltött felületeket generálni. A Hiba oka az, hogy csak a teli színnel történt kitöltéseket képes felismerni. A 24. ábrán bemutatott térkép webes változatán látszik, hogy a kontúr vonalakat legenerálta, viszont a megyéket „üresen”, kitöltetlenül hagyta.

25. ábra: A modul által generált Magyarország népsűrűségét bemutató térkép

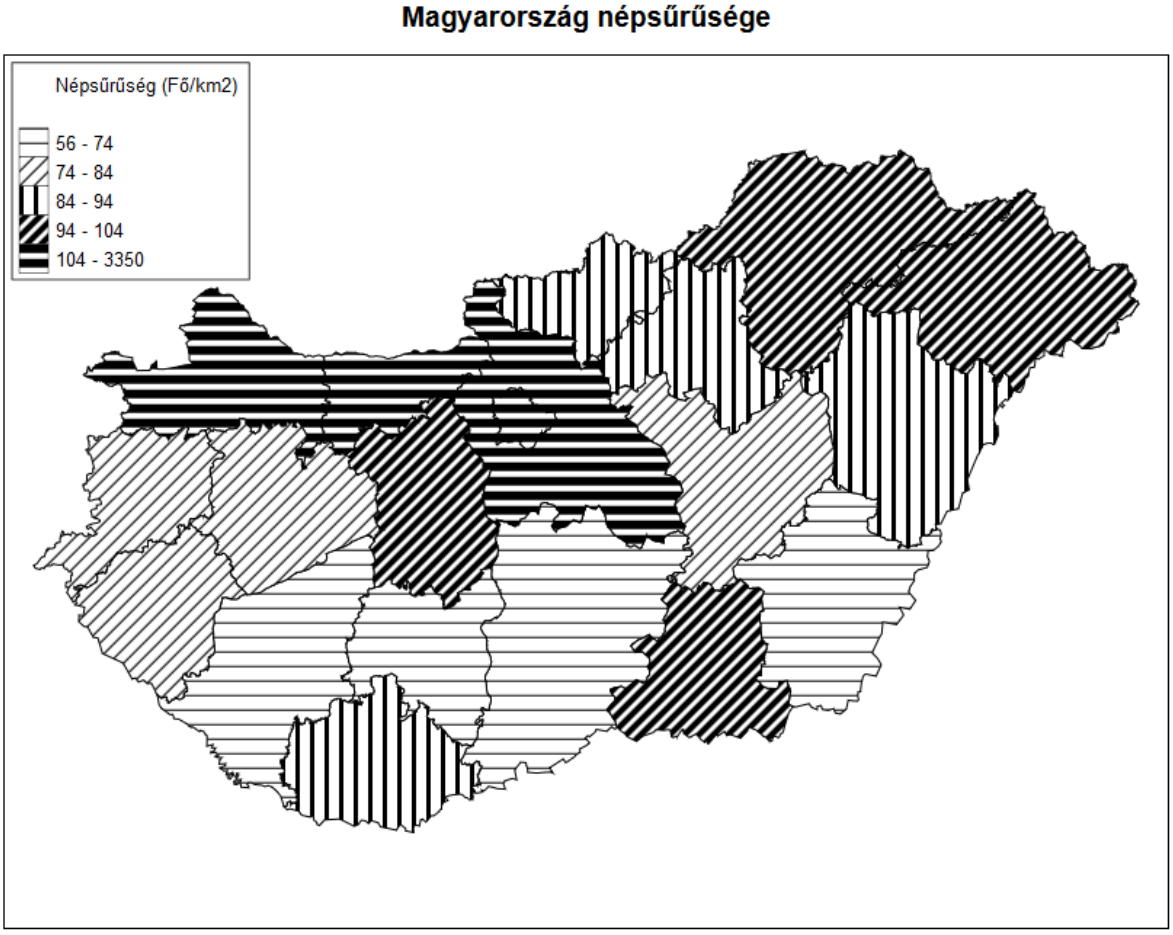
Ennek a problémának az egyik megoldása, hogy ha tovább szerkesztjük a legenerált állományokat, és hozzárendeljük az általunk használni kívánt felületi rasztert. A color.css-ben a megyékhez létrehozott színek stílusát kell megváltoztatni, *„fill”*-ről *„pattern”*-re, vagyis egy teli, egyszínű kitöltést kell megváltoztatni egy előre megadott mintára. Az egyszerűség kedvéért én már előre létrehozott mintákat használtam, melyek szabadon elérhetőek az interneten. A www.iros.github.io oldalon pontozott és vonalas mintázatok gazdag gyűjteménye található, melyek alkalmasak egy ilyen térkép elkészítésére. Használatához először a meglévő térkép index.html fájljában kell létrehoznunk az adott stílusú SVG mintázatot úgy, hogy megadjuk a minta méretét illetve elérési útvonalát:

```
<svg height="10" width="10" xmlns="http://www.w3.org/2000/svg"
version="1.1">
<defs> <pattern id="diagonal" patternUnits="userSpaceOnUse"
width="10"height="10">
<image
xlink:href="
cvMjAwMC9zdmcnIHdpZHRoPScxMCcgaGVpZ2h0PScxMCc+CiAgPHJlY3Qgd2lkdGg9JzEwJyBoZW
lnaHQ9JzEwJyBmaWxsPSd3aGl0ZScvPgogIDxwYXRoIGQ9J00tMSwxIGwyLC0yCiAgICAgICAgIC
AgTTAsMTAgbDEwLC0xMAogICAgICAgICAgIE05LDExIGwyLC0yJyBzdHJva2U9J2JsYWNrJyBzdH
Jva2Utd2lkdGg9JzEnLz4KPC9zdmc+Cg==" x="0" y="0" width="10" height="10">
</image> </pattern> </defs> </svg>
```

Ez a kód a fent említett weboldalról kimásolható. Az így létrehozott mintát hozzá kell rendelni egy, a megyék kitöltéséhez tartozó stílushoz a color.css-ben:

```
.l0r0 {stroke: #000000; stroke-width: 1; stroke-opacity: 1.0; stroke-
dasharray:;background-image:
url("
C9zdmcnIHdpZHRoPScxMCcgaGVpZ2h0PScxMCc+CiAgPHJlY3Qgd2lkdGg9JzEwJyBoZWlnaHQ9J
zEwJyBmaWxsPSd3aGl0ZScgLz4KICA8cmVjdCB4PScwJyB5PScwJyB3aWR0aD0nMTAnIGhlaWdod
D0nMScgZmlsbD0nYmxhY2snIC8+Cjwvc3ZnPg==");

        background-repeat: repeat; fill: url(#horizontal-stripe-1) }
```

Jelen példában a „horizontal-stripe-1” elnevezésű mintát használtam, ami egy vízszintes-csíkozásnak felel meg. A color.css-ben a mintázat tulajdonságait változtatni már nem tudjuk, csak a megyék keretének tulajdonságait lehet módosítani. A térkép elkészítéséhez minden egyes csoportnak el kell készíteni a stílusát.



26. ábra: Az elkészült Magyarország népsűrűsége című térkép

Az így elkészült térkép megfelel az előzetes kritériumoknak, mégis a QGIS-ben készült térképhez képest gyengébb minőséget képvisel. A sraffozás hierarchiáját sokkal nehezebb és időigényesebb felépíteni, mint egy térinformatikai szoftverben, illetve a rétegek alá- és fölérendeltségének módosíthatóságának hiánya ronthatja a térkép olvashatóságát, értelmezését.

Mint láthatjuk, a kevesebb adatot bemutató, egyszerűbb térképet kevesebb hibával tudunk létrehozni. Az eddig elkészített térképek nagy hiányossága, hogy névrajzot egyáltalán nem jelenítettünk meg. Következő térképemnél a névrajz felvételét és megjelenítését tűztem ki célul.

Magyarország megye nevei

A harmadik térképemnél a névanyag felvételének kérdését jártam körbe. A névrajz a térbeli viszonyok sajátos ábrázolási eszköze. Elhelyezése nem egzakt geometriai ismérvek alapján történik, ezért a névrajz a térképi tartalom különleges elemének számít. A GIS szoftverekben lehetőségünk van egy adott réteghez tartozó attribútum-táblázat adatait megjeleníteni a térképen, viszont ezeknek az elhelyezése nem mindig tökéletes. A névrajzot attól függően kell felvenni, hogy milyen objektumhoz rendeljük hozzá [Klinghammer, 2010].

- Pontszerű objektum: A pont fogalmát nem matematikailag, tehát kiterjedés nélkülinek, hanem grafikailag kell értelmezni: kicsiny, kör alakú ábrázolásról van szó. A hozzátartozó névrajz elhelyezése egyszerű. Meg kell adni, hogy a szöveg a pont középpontjától milyen messze kezdődjön esetleg milyen irányban illetve, hogy hány fokra legyen elforgatva.
- Vonalas objektum: Vonalas objektumként vesszük fel a térben felületi kiterjedésű tárgyak határát, vagy a térben folytonosan változó felszínek és értékek azonos értékű pontjait összekötő vonalat. Megkülönböztetünk olyan objektumot is, melyeket jelölhetünk vonalas objektumként is illetve felületként is (pl.: folyók). A névanyag megjelenítése nehéz, a térinformatikai szoftverekben sem találtam hibátlan megoldást erre a problémára. Az első lehetőség, ha megvizsgáljuk az adott objektum két végpontját, majd a vonal felező pontjához, mint egy pontszerű objektumhoz illesztjük a feliratot. Ezzel a probléma az, hogy nem tartjuk be az általános névelhelyezési szabályokat, miszerint például egy folyóra a folyásirányának megfelelően kell felrakni a nevet. Második lehetőség, ha keresünk egy hosszú szakaszt és arra ráillesztjük a nevet. Ez megint nem tökéletes megoldás, mert elképzelhető, hogy nincs olyan hosszú egyenes szakasz ahova kiférne a név, illetve nem vesszük figyelembe a névrajz többi részét, így előfordulhatnak hogy fedésbe kerülnek egymással. Összegzésként elmondhatjuk, hogy a legszebb és legjobb

megoldás, ha kézzel, egyesével rakjuk fel a vonalas objektumokhoz tartozó névrajzot, vagy áthidaló megoldás lehet a pop-up ablakok használata.

- Felületek: Ennél a megjelenítésnél az objektum térkép területe egyenletes árnyalattal kitöltött, valamilyen határvonallal lehatárolt terület. Névrajz elhelyezésére a legegyszerűbb megoldás, ha megkeressük az adott objektum centroidját (súlypontját), majd a pontszerű elemek megírásához tartozó elvvel helyezzük el a névrajzot.

QGIS-ben a névkiíratás gyorsan megoldható, a réteg tulajdonságainál a *„label”* fülön kiválasztjuk, hogy melyik attribútumot szeretnénk megjeleníteni. Miután a névrajzzal ellátott térképet a d3 Map Renderer modullal legeneráltuk láthatjuk, hogy a névrajzot elhagyja, nem jeleníti meg.

d3 JavaScript függvénykönyvtárral készített térkép

A térkép generálása során megkönnyíthetjük a dolgunkat, ha az elsődleges kulcsnak a megye neveket tartalmazó attribútumot választjuk ki. Ebben az esetben a létrejött geoJSON fájlok tartalmazni fogják a megye neveket is. Egy ilyen geoJSON fájl részlete:

```
{ "type": "Feature", "properties": { "Megye_nev": "Bács-Kiskun", "d3Css":
"l0r0" }, "geometry": { "type": "Polygon", "coordinates": [ [ [
20.097437282482385,…
```

A feladat megoldásának az első lépése, hogy létrehozunk egy stílust, mellyel a névrajzot szeretnénk megjeleníteni. A változatosság miatt itt nem a color.css-ben, hanem a HTML <head> címkéjében hoztam létre a stílust:

```
.nevrajzstilus {
            fill: black;
            fill-opacity: .8;
            stroke-width: 0px;
            font-size: 10pt;
            font-family: arial;
            text-anchor: middle;
          }
```

A következő lépés, hogy létrehozunk egy üres csoportot.

```
var vectors4 = vectors.append("g"); //Településnév
    var vector4 = void 0;
```

Az így létrejött üres csoportot fel kell tölteni a kiíratni kívánt névrajzzal. A geoJSON fájlt külön nem kell meghívni, mivel a poligonok kirajzoltatása miatt alapból be van hívva. Az üres csoport feltöltése:

```
vector4=vectors4.selectAll(".nevrajzstilus").data(object0.features)
;
vector4.enter()
.append("text")
.attr("class", function(d) { return "nevrajzstilus " + d.Megye_nev;
})
.attr("transform", function(d) { return "translate(" +
path.centroid(d) + ")"; })
.text(function(d) { return d.properties.Megye_nev; });
```

A kód első sorában megadjuk a használni kívánt stílust, illetve hogy melyik geoJSON fájlban keresse az adatokat. Az *„append*" után meg kell adnunk, hogy szöveget (text) szeretnénk megjeleníteni. Hozzárendeljük az adatokat tároló attribútum nevét (Megye_nev) majd definiáljuk, hogy hogyan szeretnénk elhelyezni a névrajzot. A *„path.centroid(d)"* sorral adjuk meg, hogy keresse meg a felület súlypontját, majd arra ráhelyezze a felirat közepét.

A feladat utolsó része, hogy beállítsuk a nagyítás utáni betűméretet.

```
vector4.style("font-size", 13 / d3.event.scale+"pt");
```

Az előző fejezetekben bemutatott eljáráshoz képest annyi a változás, hogy vonalvastagság helyett a betűk méretét kell megadni (*„font-size"*). A kód csak akkor fog helyesen lefutni, ha a sor végén megadjuk a betűméret mértékegységét is. (*„pt"*)

27. ábra: Az elkészült Magyarország megye nevei című térkép

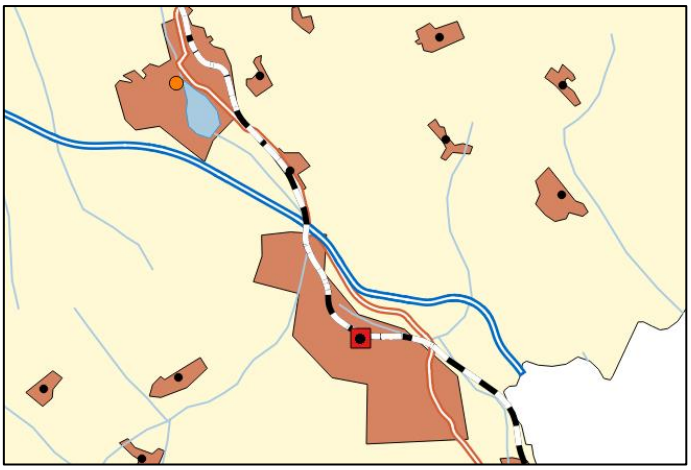
Az így elkészült térkép megegyezik a QGIS-ben készült térkép minőségével. Elkészítése nem tart sokáig, viszont ha sok megírást szeretnénk felvenni a térképre, akkor kisméretarányban a névrajz „összecsúszását” nem fogjuk tudni elkerülni.

Komárom-Esztergom megye települései

Egyes tematikus térképeknél felmerülhet az igény a települések felvételére. Ezt megtehetjük alaprajzszerűen, vagy különböző jelekkel is. A következő térképemnél Komárom-Esztergom megye településeinek ábrázolását tűztem ki célul úgy, hogy a települések jogállásuk szerint legyenek csoportokba osztva. További célom volt, hogy olyan adatsorokat is megjelenítsek, amelyeket az eredeti shape fájl nem tartalmazott. Az adatok megjelenítése pop-up ablakok segítségével oldottam meg.

QGIS-ben a kategorizált jelek megjelenítésére nagy szabadságot kap a térképkészítő. Az adott réteg tulajdonságainál a *„stílus”* fülön a *„kategorizált”* megjelenítési formát kell kiválasztani, majd ki kell választani a használni kívánt adatsort. Ezután választhatunk az előre

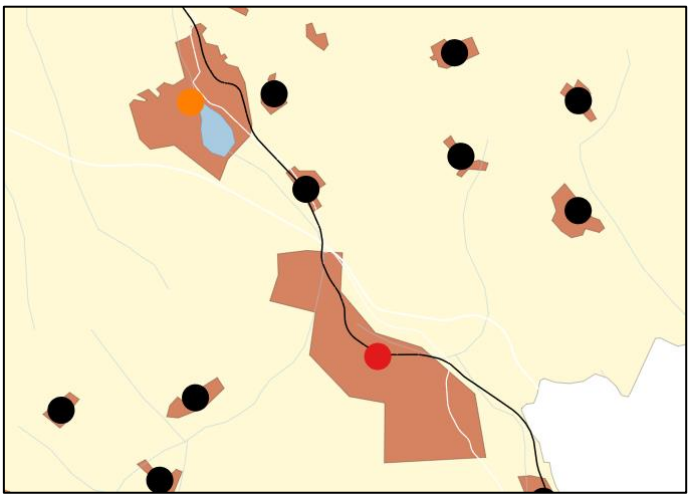
megadott jelek közül, vagy akár mi is létrehozhatunk új jeleket. A programban lehetőségünk van különféle SVG szimbólumok használatához is.



28. ábra: Kategorizált település megjelenítés QGIS-ben

A modul sikeres lefutása után láthatjuk, hogy az elkészült térkép számos hibával rendelkezik:

1. A városok jelölésére szolgáló jelek formája független az eredeti jelektől. Minden szabályos forma esetén kört generál a modul. Az eredeti jelből csak a kitöltés színét tartja meg. Ha valamilyen SVG szimbólummal próbálunk egy objektumot jelölni, akkor a modul törölni fogja az adott objektumot.
2. A jelmagyarázatot minden esetben egyszínű kitöltéssel rendelkező négyszögekből áll, így pontszerű, illetve összetett objektumok megjelenítésére nincs lehetőségünk.
3. A vasútnak csak az alsó rétege kerül ábrázolásra.
4. Vonalas elemeket elvékonyítva jeleníti meg.



29. ábra: A modul által generált jelkulcs

Ebben a fejezetben az első problémára keresek megoldást, a többi pontban felsorolt hibák kezelését az előző fejezetekben ismertettem.

Kategorizált jelek megjelenítése

A modul jelenlegi verziója az egy színű kitöltésű pontszerű szimbólumon kívül mást nem tud generálni. Valószínűleg ha befejezik a fejlesztést, akkor ezt javítani fogják. A d3.JS függvénykönyvtár a következő előre definiált jeleket képes megjeleníteni:

- Kör (Circle)
- Kereszt (Cross)
- Rombusz (Diamond)
- Négyzet (Square)
- Háromszög (triangle-up)
- Fordított háromszög (triangle-down)

A modul által generált kódsor meghívja a koordinátapárokat tartalmazó geoJSON fájlt, illetve megadja a használni kívánt megjelenítési stílust.

```
vector8 = vectors8.selectAll("path").data(object8.features);
      vector8.enter()
        .append("path")
        .attr("id", function (d) { return d.properties.ksh_kod; })
        .attr("d", path)
        .attr("class", function (d) { return d.properties.d3Css;
});
```

Ahhoz hogy a kívánt megjelenítési formát elérjük, az alapjaitól kell átalakítani a kódot.

- Meg kell hívnunk a használni kívánt szimbólumot, jelen esetben a négyzetet:

```
ad3.svg.symbol().type("square")
```

  Ha új szimbólummal dolgozunk, akkor a koordinátáktól függetlenül minden egyes jelet a térkép bal felső sarkában (0 , 0) helyezi el. Emiatt hozzá kell adnunk egy transzformációt a kódhoz, melyben megadjuk, hogy melyik jelet melyik koordinátapár szerint transzformálja. A felhasznált geoJSON fájlban minden koordináta pár egy kétdimenziós tömbben van. Ha hivatkozni szeretnénk a koordinátákra, akkor a szélességi adatot (Y) az első tömb [0] első elemével [0], a hosszúsági adatot (X) az első tömb [0] második elemével [1] kapjuk meg.

```
d.geometry.coordinates[0][0],d.geometry.coordinates[0][1]
```

- Ha különböző kategóriába sorolt jeleket különböző méretekkel szeretnénk megjeleníteni, akkor meg kell adnunk egy elágazást a kódunkban. Jelen esetben a legjobb megoldás, ha az *„if-else”* szerkezetet használjuk. Minden stílushoz rendeltem egy adott méretet, így például ha az adott jel színe piros (*„l8c0”*), akkor a mérete (*„scale”*) legyen 1.0 pont.

A fentiek alapján elkészített kódsor:

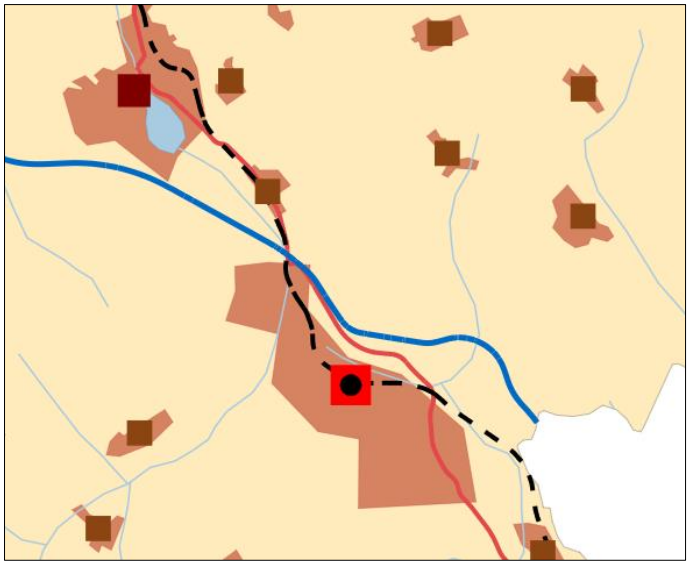
```
//Alap adatok megadása:
vector8 = vectors8.selectAll("point").data(object8.features);
vector8.enter()
.append("path")
.attr("id", function (d) { return d.properties.KSH_kod; })
.attr("d", d3.svg.symbol().type("square"))

//Transzformáció, illetve a stílus szerinti különböző méretek
megadása:
.attr("transform", function(d) {if ( d.properties.d3Css=="l8c0")
{return "translate(" +
projection([d.geometry.coordinates[0][0],d.geometry.coordinates[0][
1]]) + ") scale(1.0)";}
          else if  ( d.properties.d3Css=="l8c1") {return
"translate(" +
projection([d.geometry.coordinates[0][0],d.geometry.coordinates[0][
1]]) + ") scale(1.6)";}
          else {return "translate(" +
projection([d.geometry.coordinates[0][0],d.geometry.coordinates[0][
1]]) + ") scale(1.3)";}       })
          .attr("class", function (d) { return d.properties.d3Css;
})

//”click” esemény létrehozása:
.on("click", function (d) { return showTip(d.properties.KSH_kod);
});
```

Az így elkészült jelek már majdnem megfelelnek a QGIS-ben létrehozott jelkulcsnak, már csak a megyei jogú városokhoz tartozó összetett jelet kell elkészíteni. A piros négyzet alapja már megvan a jelnek, csak egy fekete pontot kell elhelyezni a közepére. Ennek az elkészítésének az első lépése, hogy létrehozunk egy üres csoportot, melyet feltöltjük a települések adataival és megadjuk neki, hogy kör (*„circle*”) szimbólummal jelenítse meg a térképen. A kódunk még nincsen kész, mert így minden településjelhez rendel egy fekete pontot, ezért egy kikötést

kell tenni, miszerint csak akkor jelenítse meg ezt a jelet, ha a megyei jogú városhoz rendelt stílussal (*„l8c1”*) találkozik.

```
//Alapadatok megadása:
vector9 = vectors9.selectAll("point").data(object8.features);
vector9.enter().append("path")
.attr("id", function (d) { return d.properties.KSH_kod; })
.attr("d", d3.svg.symbol().type("circle"))

//Transzofmráció, illetve a kikötés megadása

.attr("transform", function(d)
{if  ( d.properties.d3Css=="l8c1") {return "translate(" +
projection([d.geometry.coordinates[0][0],d.geometry.coordinates[0][
1]]) + ") scale(0.75)";}
          else {return "translate(" +
projection([d.geometry.coordinates[0][0],d.geometry.coordinates[0][
1]]) + ") scale(0.0)";}       })
attr("class", "point")

//”klikk” esemény meghívása
.on("click", function (d) { return showTip(d.properties.KSH_kod);
});
```

Hibája a kódnak, hogy különböző típusú jeleket egyszerre egy geoJSON fájlban nem tudjuk megjeleníteni. Ha pontosan szeretnénk lemásolni a QGIS-ben készült jelkulcsot és több fajta szimbólumot szeretnénk használni, akkor minden egyes csoportba tartozó jelet külön geoJSON fájlban kell tárolnunk. Ebben az esetben rétegenként eltudjuk végezni az előbb bemutatott beállításokat.



30. ábra: Átalakított jelkulcs

Adatok feltöltése plusz adatsorokkal

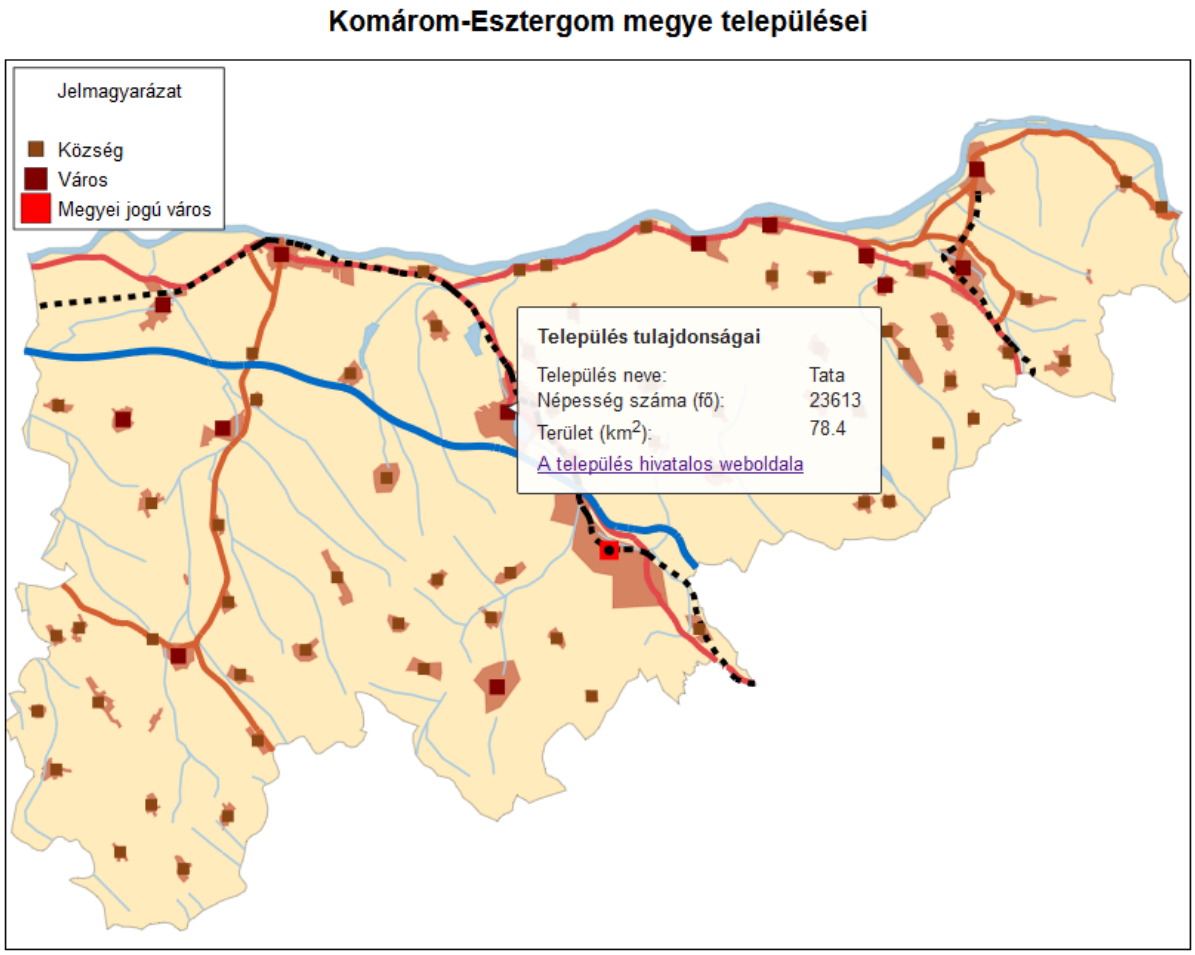
A pop-up ablakok használata lehetővé teszi, hogy több adatot is megjelenítsünk egy-egy településhez. A pop-up ablak az info.csv fájlból veszi ki az adatok, melyeket a HTML fájl <table> vezérlő szerkezetben tudjuk meghívni. Az térképhez tartozó info.csv fájl felépítése (részlet):

```
Telnev,KSH_kod,Nepesseg,Terulet,honlap
Aka,06682,243,18.1,http://www.aka.hu/
Almásfüzitő,32346,2047,8.19,http://www.almasfuzito.hu/
Ács,04428, 6071, 103.83,http://www.acs-varos.hu/
Ácsteszér,18139,696,17.71,http://www.acsteszer.hu/
```

Jelen példában négy adatot szerettem volna megjeleníteni: A település nevét, népességét, területét illetve a hivatalos honlapját. A HTML fájlban a következő képen hívtam meg:

```
<table>
<tr><td><b>Település tulajdonságai</b></td><td>
<tr><td>  </td></tr>
<tr><td>  </td></tr>
<tr><td>Település neve:  </td><td>{Telnev}</td></tr>
<tr><td>Népesség száma (fő):  </td><td>{Nepesseg}</td></tr>
<tr><td>Terület (km<sup>2</sup>):  </td><td>{Terulet}</td></tr>
<tr><td>  </td></tr>
<tr><td><a href="{honlap}"> A település hivatalos weboldala</a>
</td></tr>
</table>
```

A {} zárójelek közé kell megadni az info.csv fájlban található oszlopok neveit. A település weboldala hiperhivatkozásként jelenik meg.



31. ábra: Az elkészült Komárom-Esztergom megye települései című térkép

Az elkészült térkép szép és igényes, a pop-up ablakban pedig akár nagy adatsorok megjelenítése is lehetséges. Hátránya, hogy ha különböző szimbólumokat szeretnénk használni, akkor elkészítése nagy időráfordítást igényel. A jelmagyarázat ennél a térképnél is kritikus pontnak számít, ugyanis összetett jeleket itt se tudunk megmagyarázni.

Magyarország működő bányáinak száma 1970 és 2005 között

A következő célom a d3 Map Renderer modul diagram funkciójának a bemutatása volt. A d3.JS függvénykönyvtár rengeteg összetett diagram típust támogat, ugyanakkor a kartográfiában általában nem célszerű túl összetett típusokat használni.

A térkép tematikája a Magyarországon működő bányák számainak az alakulása 1970 és 2005 között. Ez idő alatt összesen négy nagyobb mértékű bányabezárás volt: 1974-,1984-,1995- és 2005-ben. Térképemmel települések szerint mutatom az adott időszakok változásait. Az adatok forrása az Országos Területfejlesztési és Területrendezési Információs rendszer (www.teir.hu).

Modul beállítása

Miután elkészült a QGIS-ben az állomány, a modul *„Viz”* fülén tudjuk a diagramhoz tartozó beállításokat elvégezni. Első lépésben meg kell adnunk, hogy milyen fajta diagramot szeretnénk használni. Számos lehetőség közül választhatunk, jelen példában a vonal típusú *(„line-chart”)* típust választottam. Következő lépésben megadhatjuk a diagramot tartalmazó pop-up ablak, ezzel a diagram méretét, majd a megjeleníteni kívánt adatokat. Itt csak a fő réteg numerikus elemeket tartalmazó mezői közül tudunk választani. Az itt beállított adatokkal a grafikon X tengelyét töltjük fel. Figyelni kell arra, hogy az adatok megjelenítési sorrendje megegyezik a behívási sorrenddel. Az Y tengelyhez tartozó adatokat a modul automatikusan generálja az X tengelyhez tartozó adatok minimum és maximum értékei alapján. A kiválasztottuk az adatokat a „+” jellel tudjuk hozzáadni a diagramhoz, az adatsor címének megadásával. Az ablak alján található *„X-axis label”* sornál tudjuk megadni az X tengelyhez tartozó megírásokat. A megírás csak számokat tartalmazhat, szóközökkel elválasztva.

Az általam használt modul 0.10.3-as verzió csak az *„integer”* típusú számokkal képes dolgozni, a 0.10.5-ös verzióban ezt már javították, és a QGIS-ben használt *„integer64bites”* számábrázolást is felismeri.

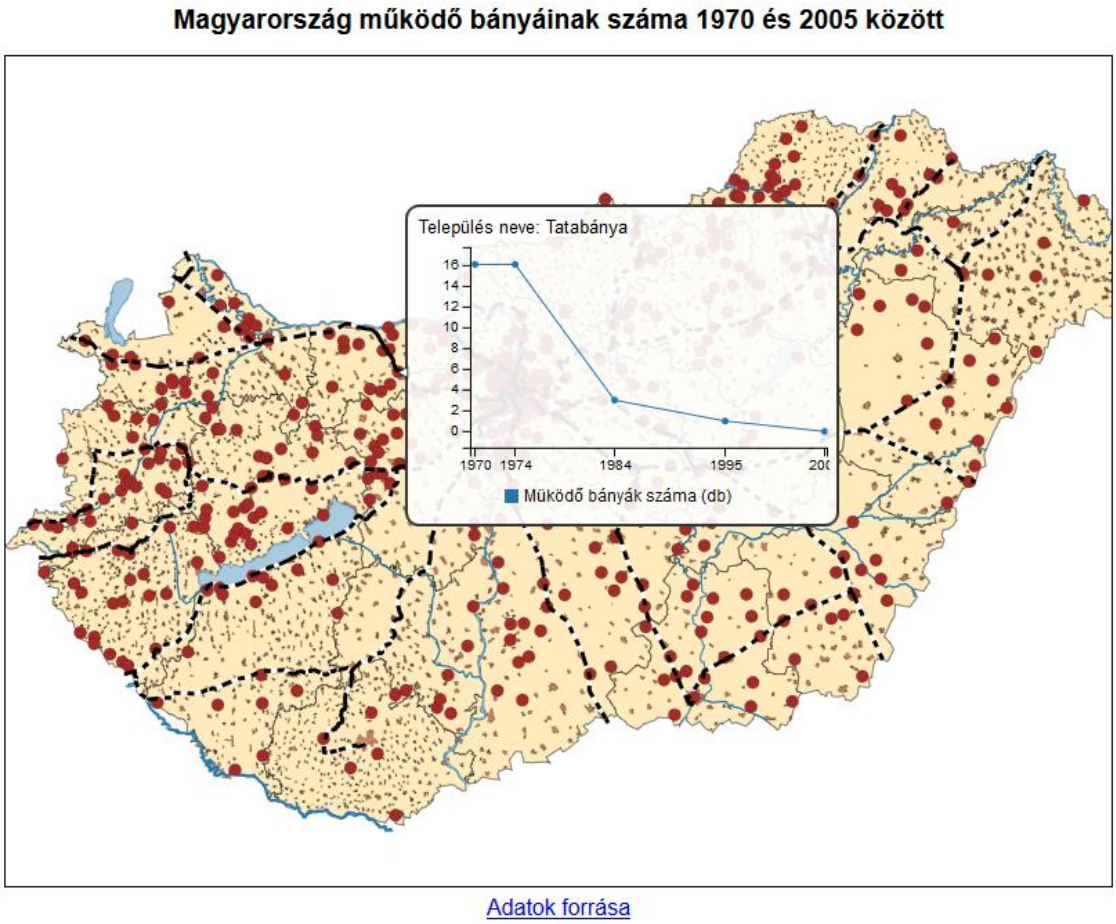
A beállítások elvégzése után az *„OK”* gomb lenyomásával tudjuk legenerálni a térképet.

Az elkészült térképet az előző fejezetekben leírtak szerint formáztam. A diagramban feltüntetett adatok tárolása ugyanúgy történik, mintha csak sima adatsorokat szeretnénk megjeleníteni pop-up ablakban. Az elkészült diagram tulajdonságai a tip.ccs-ben, az c3min.ccs-ben és az index.html-ben tudjuk módosítani. A diagram leírása a HTML fájlban:

```
function chart(obj){
      var par = d3.select(".d3-tip #chart")
//X tengely megírásai

      var labels = ["x", 1970,1974,1984,1995,2005,];

//megjelenített adatsorok nevei
      var data1 = ["data1",
obj["all_int"],obj["1974_banya"],obj["1984_banya"],obj["1995_banya"
],obj["2005_banya"]];
      var chart = c3.generate({
        bindto: par,
        data: {
```

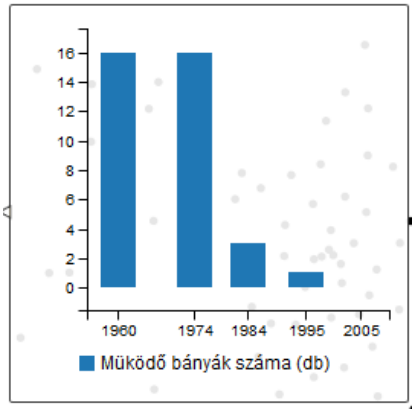
```
            x: "x",
            type: "line", //diagramtípus
            columns: [
              labels,data1
            ],

  //diagram neve és mérete
            names: { data1: "Müködő bányák száma (db)"   }
          },
          size: {
            width: 300,
            height: 200
          },

//Y tengely megírásai
          axis: {
            y: {
              min: 0,
              max: 16
            }
          }
        });
      }
```

A modul automatikusan generál egy „klikk-eseményt”, amely az adatsor nevére történő kattintásra aktiválódik. Ha rá kattintunk, akkor az adott adatsor eltűnik a diagramról. Ez a funkció több adatsor egyszerre történő megjelenítésénél teszi könnyen átláthatóvá a diagramot.



32. ábra: Az elkészült Magyarország működő bányáinak száma 1970 és 2005 között

A vonal típusú diagramnál általános hiba, hogy az X-tengelyhez tartozó megírásoknak az utolsó tagját nem jeleníti meg teljes egészében. A probléma oka valahol a JavaScript kódban lehet, erre megoldást nem találtam. Ha ugyan ezt az adatsort oszlop-diagrammal ábrázoljuk, akkor a probléma megszűnik.



33. ábra: Tatabánya működő bányáinak száma oszlop diagrammal ábrázolva

Ezen kívül a pop-up ablakokhoz tartozó szimbólumok méretezésében is észleltem egy hibát. Az index.html fájlban minden objektumoknak megadhatjuk a nagyítás utáni méretet, így különböző méretarányokban elkerülhető, hogy a jelek fedjék egymást. Ez alól kivételnek számítanak a szimbólumok, melyek dinamikus megjelenítésére nem találtam megoldást. Ez a jelenség a térképnél Budapest és környékénél okoz komoly gondot. Mivel azon a környéken számos bánya volt a múltban, a térkép ezen része értelmezhetetlenné válik.

Összességében elmondhatjuk, hogy a d3.JS-vel könnyen tudunk előre definiált diagramokat megjeleníteni, melyek ugyan nem hibátlanok, de megjelenésben szépek és akár nagy adatmennyiséget is megtudunk egyszerre gyorsan jeleníteni.

Világtérkép készítése

Az eddig elkészített térképeknél arra törekedtem, hogy általános tematikus térképeket hozzak létre. A d3.JS függvénykönyvtár azonban számos egyéb függvénnyel rendelkezik, amivel hasznos és látványos térképeket tudunk létrehozni. A könyvtár tartalmazza a legnépszerűbb vetületek adatait, ezáltal lehetőségünk van számos vetület közül választani, illetve ki is rajzoltathatjuk a fokhálózati vonalakat is.

Egy kartográfiában nem jártas ember számára nehéz elképzelni a különböző vetületek adta előnyöket és hátrányokat. Célom egy olyan térkép elkészítése volt, amelynél ábrázolom a Föld országait, a tengeráramlatokat illetve a fokhálózati vonalakat. Egy legördülő ablakban lehetőségünk van kiválasztani a használni kívánt vetületet, úgy hogy a régi vetület „átalakuljon” az új vetületté. Mind a két réteghez rendelek egy adatbázist, mely az objektumra való kattintás után kiírja az adott ország vagy áramlat nevét. Az országokat ezen kívül még elláttam egy „*mouse-over*” eseménnyel, amely hatására ha az egerünket rávisszük egy ország poligonra akkor annak kitöltése megváltozik. A következő pontokban bemutatom a térkép elkészítésének fontosabb részeit.

Tengeráramlatok elkészítése

A meglévő tengeraramlatok.shp fájl a tengeráramlatok vonalait, illetve attribútumait tartalmazta. Gondot jelentet a vonalakhoz tartozó nyilak elkészítése. A problémára nem találtam tökéletes megoldást. Az első ötletem az volt, hogy a vonalak végére elhelyezek egy pontszerű jelet, majd az index.html fájlban háromszögekké (*„triangle-up”*) alakítom át. Ez sikerült is, viszont ezeket irányba kellett forgatni ahhoz, hogy visszaadják az áramlatok

irányát. Külön-külön a nyilak forgatására megoldást nem találtam, csak ha minden nyílnak külön stílust definiálok, melyekhez egy-egy forgatási értéket rendelek. Ez hatalmas időráfordítást igényelt volna, melyet nem tartottam jó megoldásnak.

Második ötletem, hogy a nyilakat poligonként felveszem és hozzácsatolom a vonalakhoz, így kiküszöbölve a forgatást. QGIS-ben ez a módszer szépen és hibátlanul működött viszont miután legeneráltam a térképet látható volt, hogy a különböző vetületek szerint a nyilakat reprezentálni kívánt poligonokat különböző módon torzította, illetve egyes helyeken el is mozdította a helyéről. Mivel nem szerettem volna a nyilakat lehagyni a térképről ezért végül a kisebb hibával rendelkező megoldást választottam.

Két adatsor megjelenítése

Az eddig elkészített térképeknél megfigyelhető, hogy mindegyiknél csak egy réteg volt aktív, az-az egy réteghez volt adatsor rendelve, melyeket kattintás hatására valamilyen pop-up ablakban jelenítettem meg. Ennek, oka, hogy a modulban csak egy réteghez tudunk pop-up ablakot rendelni. Ennél a térképnél az ország neveket illetve a tengeráramlatok neveit is szerettem volna megjeleníteni, ezért ezt kézzel kellett megoldani.

Először elkészítettem a kontinenseket tartalmazó állományt, majd utána külön a tengeráramlatok állományt. A használni kívánt részeket egy állományba másoltam át. A tengeráramlásokhoz tartozó .csv fájlt át kellett nevezni, hogy elkerüljük az adatok ütközését. Az index.html fájlban létre kell hozni egy üres csoportot, amit fel kell tölteni az átmásolt geoJSON adataival.

Mouse-over esemény

Minden programozási nyelvben lehetőségünk van eseményeket létrehozni. Ez azt jelenti, hogy az objektumok bizonyos eseményeket csak a felhasználó kérésére végeznek el. Ilyen kérés lehet, például ha rákattintunk, vagy ha rávisszük a kurzort egy objektumra. Ezek közül a pop-up ablakok automatikusan meghívták a *„click”* eseményt, mely segítségével tudjuk megjeleníteni az ablakokat. A térkép könnyebb értelmezése miatt azt szerettem volna elérni, hogy ha rávisszük a kurzort egy országra, akkor az egész ország területének a színe megváltozzon, majd ha megszűnik a kijelölés, akkor változzon vissza az eredeti színre. Ezt a *„.mouseover”* és a *„.mouseout”* esemény meghívásával tudtam elérni [Juhász-Kiss, 2004].

```
.on("mouseover", function(d) {  return   d3.select(this).style("fill",
"#BDBDBD");})
.on("mouseout", function(d) { return d3.select(this).style("fill",
d.d3Css)});
```

Minden eseményt az *„.on”* kezdéssel kell meghívni.

Fokhálózat kirajzoltatása

Az index.html fájlban lehetőségünk van a fokhálózati vonalak kirajzoltatására a függvénykönyvtár segítségével. Első lépésben létre kell hoznunk egy stílust a fokhálózati vonalaknak:

```
.graticule {
  fill: none;
  stroke: #6E6E6E;
  stroke-width: 0.5px;
}
```

Ezután, a vetület definiálásánál meg kell adnunk, hogy milyen sűrű fokhálózatot szeretnénk felvenni a térképre:

```
var graticule = d3.geo.graticule().step([10,10]);
```

Majd végezetül ki kell rajzoltatni a fokhálózati vonalakat:

```
vectors.append("path")
    .datum(graticule)
    .attr("class", "graticule")
    .attr("d", path);
```

A művelet hibája, hogy ezekkel a beállításokkal a fokhálózati vonalakat kicsinyítve helyezi el a térképlap közepére, ezért meg kell adnunk egy nagyítási értéket. A szükséges nagyítási értéket nem lehet egzaktul meghatározni, ezért csak közelíteni lehet a fokhálózati vonalak pontos helyzetükhöz. Ez csak a kezdeti nézetet befolyásolja, később, az első vetületváltás után a fokhálózat a „helyére került”.

Megjelenített vetületek

A felhasznált vetületek kiválasztásánál figyelembe vettem a felhasználható vetületek listáját illetve olyanokat igyekeztem kiválasztani, melyek jól bemutatják a különböző mértékű vetületi torzulásokat mind a felületeken, mint a vonalakon. A vetületek közül egy legördülő listából tudjuk kiválasztani a használni kívántat. Választható vetületek:

- Aitoff: Általános torzulású egyéb képzetes vetület

- Mercator: Szögtartó valódi hengervetület
- Eckert I: Általános torzulású képzetes hengervetület
- Eckert III: Általános torzulású képzetes hengervetület
- Eckert V: Általános torzulású képzetes hengervetület
- Loximuthal: Általános torzulású képzetes hengervetület
- August: Általános torzulású képzetes kúpvetület
- Lagrange: Szögtartó képzetes kúp vetület
- Albers: Területtartó valódi kúpvetület
- Bonne: Területtartó Igazi képzetes kúpvetület
- Hammer: Területtartó egyéb képzetes vetület
- WinkelTripel: Általános torzulású egyéb képzetes vetület

Az index.html fájlban a következő kódsorral tudunk egy-egy vetületet meghívni a függvénykönyvtárból:

```
var options = [
{name: "Aitoff", projection: d3.geo.aitoff()}, ]
```

Vetületválasztás

A behívott vetületek külön-külön már megtudjuk jelenteni, de azt szeretnénk elérni, hogy a felhasználó egy legördülő ablakban kitudja választani a használni kívánt vetületet. *„Vetuletvaltas”* néven létrehoztam egy új függvényt, amely megvizsgálja a legördülő sávban kiválasztott névhez tartozó vetületet, majd ahhoz képest módosítja a térképi vetületet. A d3.JS függvénykönyvtár rendelkezik egy függvénnyel, mely lehetővé teszi, hogy a két vetület ne csak kicserélje egymást, hanem az egyik „átváltozzon” a másikká. Ez a *„projectionTween”* függvény. A függvény meghívás előtt az általános beállításokat kell elvégezni, mint például, hogy az „átváltozás” mennyi idő alatt menjen végbe.

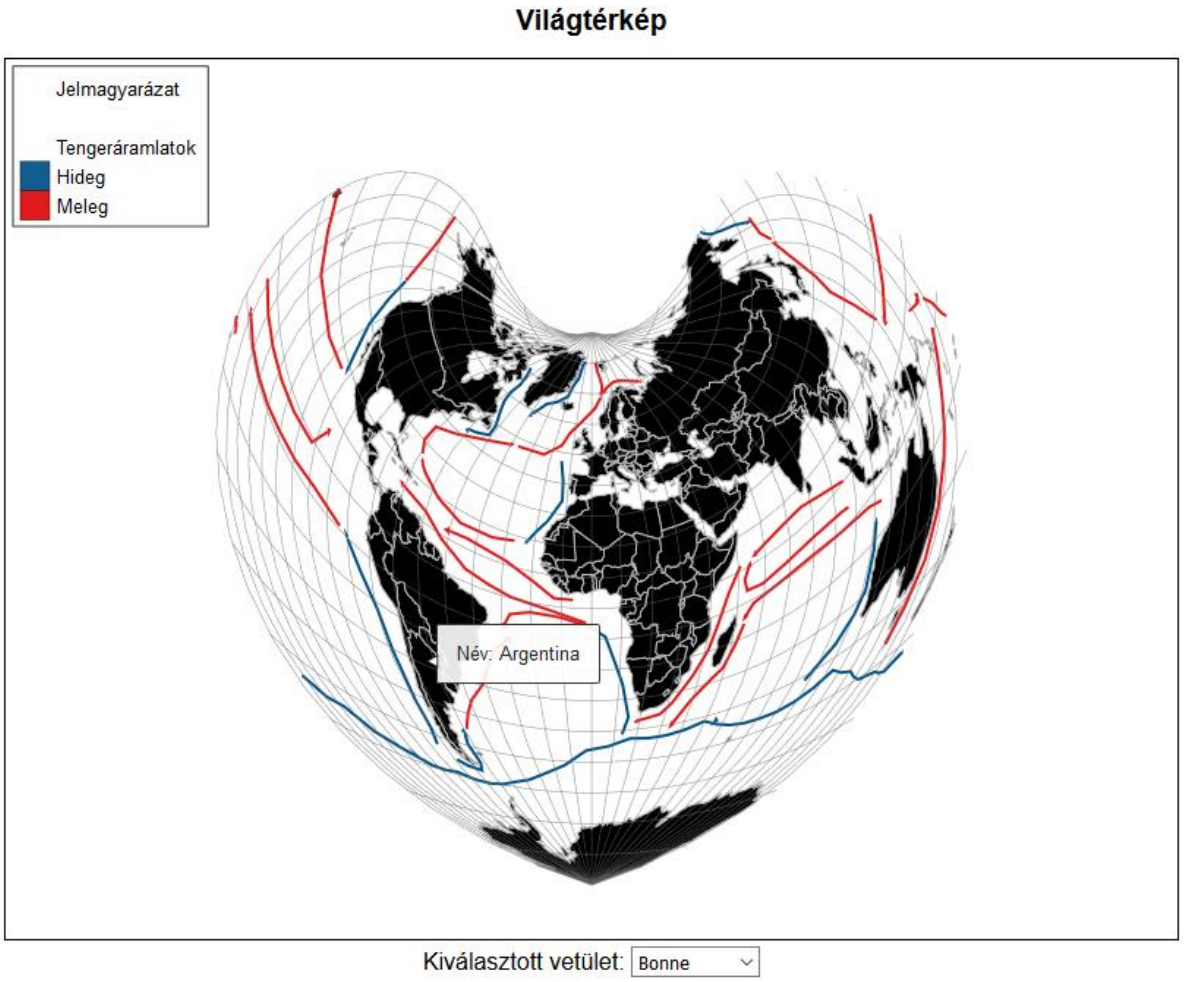
```
function vetuletValtas(vetuletNeve) {//Vetületváltás
  var i=0;
var options = [
    {name: "Aitoff", projection: d3.geo.aitoff()},
    .
    . //Összes megjeleníteni kívánt függvény felsorolása
];
    for (i in options) {
          if (vetuletNeve==options[i].name) {

          svg.selectAll("path").transition()
```

```
                .duration(1500)
                .attrTween("d", projectionTween(projection,
projection = options[i].projection));
            }
        }
      }
```

A projectionTween függvény felépítése:

```
function projectionTween(projection0, projection1) {
  return function(d) {
    var t = 0;

    var projection = d3.geo.projection(project)
        .scale(1)
        .translate([width / 2, height / 2]);

    var path = d3.geo.path()
        .projection(projection);

    function project(λ, ϕ) {
      λ *= 180 / Math.PI, ϕ *= 180 / Math.PI;
      var p0 = projection0([λ, ϕ]), p1 = projection1([λ, ϕ]);
      return [(1 - t) * p0[0] + t * p1[0], (1 - t) * -p0[1] + t * -
p1[1]];
    }

    return function(_) {
      t = _;
      return path(d);
    };
  };
}
```

A függvény használata, illetve a teljes hozzá tartozó kódsor megtalálható a https://bl.ocks.org/ weboldalon.



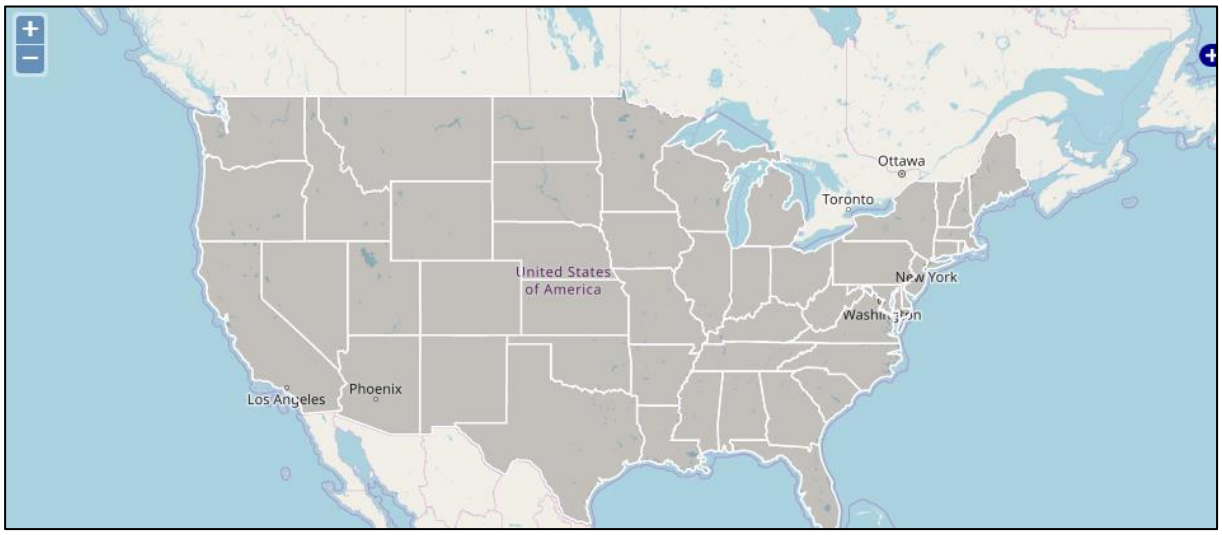
34. ábra: Az elkészült Világtérkép, Bonne vetületben ábrázolva

Az elkészült térkép nem csak látványos, de tanulságos is. A kartográfiában nem jártas emberek is – akik többségében csak a szögtartó Mercator vetülettel találkoznak – láthatják, hogy milyen nagymértékű torzulások léphetnek fel egy-egy vetület esetén. A megjelenítés itt sem tökéletes: többek között a nyilak torzulása, a vonalak töredezettsége és a fokhálózat pontatlan megjelenítése. Ezek ellenére az ilyen fajta megjelenítésnél tudjuk igazán kihasználni a d3 JavaScript függvénykönyvtár előnyeit.

9. d3 JavaScript függvénykönyvtár beágyazása más rendszerekben

A d3.JS függvénykönyvtár elterjedésével egyre több felhasználási lehetőséget fejlesztenek ki hozzá. A dolgozatomban csak olyan térképek elkészítését mutattam be, ahol az alaptérképet is én készítettem el. Felhasználása viszont úgy is lehetséges, hogy már meglévő alaptérkép állományt használunk fel.

Egyik ilyen párosítás lehet a d3.JS és az OpenLayers párosítása. Alapját a közös fájlformátum, a topoJSON adja. Használata első sorban azért lehet majd előnyös, mert bonyolult grafikonok, és különböző dinamikus tartalmak megjelenítése lehetségessé válik. A függvénykönyvtár ilyen irányú felhasználása egyelőre még nem kiforrott, részletes dokumentáció nem készült hozzá, csak néhány példa, amely elérhető az OpenLayers weboldalán.



35. ábra: OpenLayers-be ágyazott d3.JS függvénykönyvtár használata[9]

[9] A térkép forrása: http://bl.ocks.org/mbertrand/5218300

10. Összegzés

Diplomamunkámban röviden áttekintettem a mai webszerkesztéshez használt programozási nyelvek, és formátumok alapjait. Részletesen bemutattam a QGIS d3 Map Renderer modul felépítését, működését és az általa létrehozott fájlokat. Ismertettem a d3 JavaScript függvénykönyvtár felhasználási lehetőségeit a kartográfiában, bemutatva az előnyeit és a hátrányait is. Összesen hat darab térképet készítettem, ahol különböző tematikus ábrázolási módszereket mutattam be.

Összességében elmondható, hogy ha adatbázis alapú dinamikus webtérképet szeretnénk készíteni, akkor a d3.JS függvénykönyvtár használata egy jó és innovatív megoldás lehet. Ahogy az elkészített térképeimen bemutattam, inkább a kevesebb adatot megjelenítő, de sokkal interaktívabb térképek készítése az ajánlott. A függvénykönyvtár segítségével képesek vagyunk feliratot, különböző felületi rasztereket, pop-up ablakokat és grafikonokat megjeleníteni. Fő előnye, az előre megírt függvények használatában rejlik. Ezek segítségével olyan interaktív térképeket tudunk létrehozni, melyek kategóriájukban kimagasló minőséget képviselnek.

A módszer adta lehetőségek teljes körű kihasználásához a felhasznált programozási nyelvek mély szintű ismerete szükséges. Negatívumnak számít, hogy egy-egy térkép elkészítése –más módszerekhez képest– kifejezetten időigényessé is válhat. Dolgozatomban számos kisebb és nagyobb hibára felhívtam a figyelmet, mind a modul, mind a függvénykönyvtár kapcsán. Ezek közül talán a leginkább égetőbb probléma a jelmagyarázat alakíthatóságának hiánya.

Hibái ellenére fontos tényező, hogy mind a modul, mind a d3.JS még fejlesztés alatt áll, ezért valószínűsíthető, hogy a jövőben számos hiányosságot ki fognak javítani, amely akár a függvénykönyvtár nagymértékű elterjedését jelentheti a web-kartográfia világában.

11. Irodalomjegyzék

Clive Cookson (2013): "Edward Tufte".FT Magazine. Július 26-i szám, ISSN 0307-1766

Dr. Gede Mátyás (2012): Open Source rendszerek a térinformatikai gyakorlatban – Interaktív webtérképek készítése OpenLayers és MapServer használatával, Oktatási segédanyag, Eötvös Loránd Tudományegyetem, Informatikai Kar

Dr. Pődör Andrea (2010): Térképek a weben, Oktatási segédanyag, Nyugat-magyarországi Egyetem Geoinformatikai Kar

Ember László (2010): Webszerkesztés, a web programozás alapjai: 2. modul CSS és JavaScript programozás, Oktatási segédanyag, Budapest

ESRI, Incorporated (1998): ESRI Shapen le Technical Description. An ESRI White Paper, www.esri.com/library/whitepapers/pdfs/shapefile.pdf
(utolsó elérés: 2017.12.19.)

Google (2013): Fusion Tables, www.google.com/fusiontables
(utolsó elérés: 2017.12.18)

Huszár István (2009): HTML összefoglaló Segédlet a HTML (HyperText Markup Language) programozási nyelv alapjaihoz, a honlapok készítéséhez. Oktatási segédanyag, Budapest

Internet Standards Track Document (IETF) (2016): The GeoJSON Format, http://geojson.org/geojson-spec.html
(Utolsó elérés: 2017.12.18)

Juhász Tibor-Kiss Zsolt (2004): Tanuljunk programozni, ComputerBooks kiadó, Budapest, pp. 223-229.

Klinghammer István (2010), Térképészet és geoinformatika I., ELTE Eötvös Kiadó, Budapest, pp. 133-139.

Mike Dewar (2011): Getting Started with D3, O'Reilly Media, Inc., Sebastopol, pp. 1-14.

Nick Qi Zhu (2013): Data Visualization with D3.js Cookbook, Packt Publishing Ltd., Birmingham, pp. 1-23.

Savarese Software Research Co. (2012): Ssrc SVG: SVG Plugin for Internet Explorer, http://www.savarese.com/software/svgplugin/
(Utolsó elérés: 2017.12.17.)

Úr Bence (2011)**:** Html és css oktatás ingyenesen, *http://www.standardsmode.hu/*
(Utolsó elérés: 2017.12.18)

Simon Benten (2015): d3 MapRenderer, http://maprenderer.org/d3/
(Utolsó elérés:2017.12.19.)

12. Köszönetnyilvánítás

Szeretnék köszönetet mondani Ungvári Zsuzsannának, hogy vállalta a diplomamunkám témavezetést. Köszönöm a türelmét és az értékes tanácsait. Lelkesedése hétről hétre motivált.

13. Melléklet

A diplomamunkához mellékletként csatolt CD lemez tartalmazza:

1. A dolgozatot elektronikus formában (PDF formátum)
2. A d3 Map Renderer modult
3. Az elkészült térképek állományait, sorrendben:
    - Komarom_esztergom_megye_telepulesei: Komárom-Esztergom megye települései
    - Magyarorszag_megye_nevei: Magyarország megye nevei
    - Magyarorszag_nepsurusege: Magyarország népsűrűsége
    - Magyarorszag_uthalozata: Magyarország úthálózata
    - Mo_mukodo_banyainak_szama: Magyarország működő bányáinak száma 1970 és 2005 között
    - Vilagterkep: Világtérkép

Nyilatkozat

Alulírott **Rátvai Dániel** (NEPTUN azonosító: **YBTKVG**) a „Tematikus térképek készítése a D3.JS függvénykönyvtár segítségével” című diplomamunka szerzője fegyelmi felelősségem tudatában kijelentem, hogy dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

A témavezető által elfogadott és elbírált diplomamunka elektronikus közzétételéhez (PDF formátumban a tanszéki honlapon) **hozzájárulok.**

Budapest, 2018.január.08

……
Hallgató aláírása

Hozzájárulok a szakdolgozat benyújtásához:

Budapest, 2018.január.08

……
Témavezető aláírása