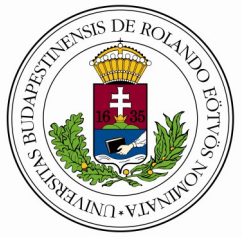
Eötvös Loránd Tudományegyetem
Informatikai Kar
Térképtudományi és Geoinformatikai Tanszék

Térben és időben változó folyamatok megjelenítése webtérképen SVG alkalmazásával

Diplomamunka

Kádár Iván
térképész mesterszakos hallgató

Témavezető:
Dr. Gede Mátyás
Adjunktus



Budapest, 2013

Tartalomjegyzék

1 Bevezetés

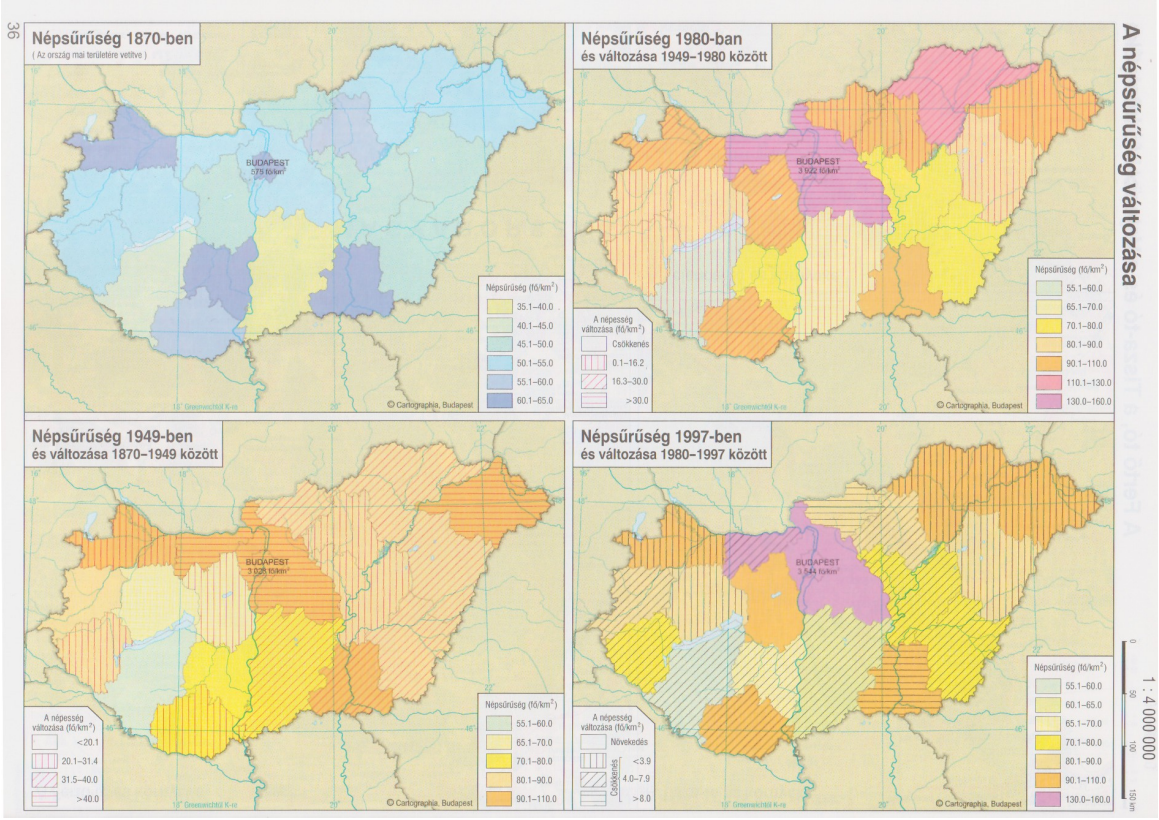
Diplomamunkám és a hozzá kapcsolódó webes alkalmazás célja, hogy bemutassam a térképi animációk megvalósításának lehetőségeit, és felvázoljak egy modern, a ma egyre inkább terjedő digitális eszközökön – mobiltelefonokon, tableteken – is megjeleníthető animációs módszert, a böngészőben történő térképrajzolást vektorgrafika (SVG) és JavaScript használatával.

Dolgozatomban először elemzem a különböző animációs lehetőségeket, megvizsgálva előnyeiket és hátrányaikat, majd röviden bemutatom az alkalmazásomhoz felhasznált technológiákat. Ezt követően bemutatom az elkészült alkalmazást, ismertetem a megvalósítás menetét, majd a kész program használatát. Végül vázolom a további fejlesztési lehetőségeket, és ismertetem a fejlesztés közben szerzett tapasztalataimat.

2 Térképi animáció készítésének és megjelenítésének lehetőségei

2.1 Előzmények

A változás ábrázolásának klasszikus megoldása, hogy azonos kivágattal elkészítünk több térképet más-más időpontról, és azokat egymás mellé téve láthatóvá válnak a különbségek. Ilyen például a Magyarország népsűrűségét ábrázoló térképsorozat a Cartographia Kft. által kiadott Középiskolai földrajzi atlasz 36. oldalán. [2.1. ábra]



2.1. ábra: A népsűrűség változása négy térképen

Ennek előnye, hogy a felhasználó alaposan megnézheti, összehasonlíthatja a térképeket, felfedezheti a változásokat.

Hátránya azonban, hogy a képek darabszáma korlátozott. Ahhoz, hogy jól összehasonlítható legyen két időpont, a térképeket egyszerre kell látnia a felhasználónak, a lapozás pedig jelentősen rontja az információ befogadását. Ezen kívül legalább a céltematikát meg kell rajzolni minden térképhez külön-külön. A korlátosság miatt jól meg kell válo-

gatni az ábrázolt időpontokat, időintervallumokat, hogy minden fontos változás megjelenjen. Például egy évszázadonként ábrázolt közigazgatási térképsorozat esetében egy-egy évszázad alatt sok határváltozás történhet. Ezek egy része esetleg nem fog megjelenni a térképeken, hogy azok tartalma áttekinthető maradjon.

2.2 Térképi animáció alapjai

A kartográfusok már az 1930-as évek elején tettek kísérleteket arra, hogy az idődimenzió felhasználásával dinamikussá tegyék a térképi ábrázolást. Már a második világháború alatt készültek filmes animációk a híradásokhoz, ahol hadmozgásokat mutattak be. (Schafer, 1993) Számítógépes animációk készítésével kísérleteztek az 1960-es években is, azonban igazán elterjedtté és népszerűvé a személyi számítógépek megjelenésével válhatott ez a lehetőség. (Peterson, 1994)

Ezzel együtt megjelentek a nyilvánvaló problémák is. Egy animáció alatt a szemlélőnek nincs elég ideje részletesen megvizsgálni az egyes képeket, így sokkal kevesebb információ tárolható egy-egy képkockán, mint egy statikus térképen.

A térképi animációnak két fajtája lehetséges, az időbeli animáció és az időfüggetlen animáció. Akkor beszélünk időbeli animációról, mikor olyan folyamatot jelenítünk meg, ami a valóságban is rendelkezik időhöz kötöttséggel. Ilyen például a népességnövekedés megjelenítése. A másik lehetőség, amikor az animáció előrehaladtával valami mást szemléltetünk, például egy adott időpontban különböző korcsoportokra vetített népsűrűségi adatok egy adott területen. (Peterson, 1996) Dolgozatomban az előbbi, tehát időbeli animációval foglalkozom, a nem időbelit csak megemlítem.

Mindkét esetben igaz, hogy míg a statikus térkép egyszerre mutatja meg minden információját, addig az animációk az idő előrehaladtával jelenítik meg információtartalmukat. Ha az animáció túl hosszúra nyúlik, a szemlélő nem tudja befogadni az egészet, mivel a rövid távú memóriája nem tárolja el a kezdeti információkat. (Harrower – Fabrikant, 2008)

Ahogy egy hagyományos térképnek van geometriai méretaránya, úgy az animációnak is van egy időbeli méretaránya. Például, ha egy tíz éves folyamatot egy percben ábrázolunk, akkor az animáció időbeli méretaránya 1:5,3 millió lesz. (Harrower – Fabrikant, 2008)

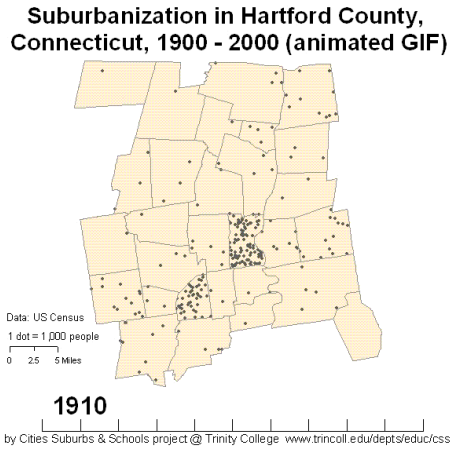
A statikus térkép jelmagyarázatához hasonlóan a térképi animációnak is szüksége van egy időbeli magyarázatra, tehát ábrázolnunk kell, hogy éppen mely időpontnál tartunk, külön-

ben a szemlélő nem fogja tudni meghatározni az időbeli méretarányt. (Peterson, 2000) Ezt a magyarázatot dinamikussá is tehetjük, ekkor a felhasználó magának választhatja ki, hogy épp mely időpontot szeretné szemlélni. (Asproth, 1995)

2.3 Mozgókép

Az animált térképek legegyszerűbb változata az elkészült önálló képeket képkockaként való összefűzése. Erre a legelterjedtebb megoldás a Graphics Interchange Format (gif) állomány alkalmazása, melynek az 1989-es, továbbfejlesztett változata (GIF89a) tette lehetővé több képkocka összefűzését. [2.2. ábra]

A formátum előnye, hogy veszteségmentes tömörítést használ, azonban ezt csak egy legfeljebb 256 színből álló paletta alapján tudja megtenni. A paletta ugyan nem előre definiált, így bármely RGB szín bele tehető, illetve lehet egy átlátszó „színt” is definiálni, azonban a 256-féle szín így is komoly korlátozásnak bizonyult a technológiai fejlődés mellett. A formátum másik problémája a zárt tömörítési algoritmus (LZW) alkalmazása volt. (Neuendorffer, 2000)



2.2. ábra: Egy képkocka egy GIF formátumú mozgó térképből

Mindezek kiváltására jött létre a Portable Network Graphics (png) fájlformátum, illetve ennek animált változata az APNG. Ennek előnye, hogy már a teljes 32 bites RGBA színskála támogatja, tehát nemcsak, hogy nem kell 256 színt kiválasztanunk, de még az egyes képpontok átlátszóságát is szabályozhatjuk 256 fokozatban. (European Commission, 2013)

A GIFnek a színkorlátozottságában is megjelenő elavultsága ellenére nagy előnye, hogy minden platformon támogatott a megjelenítése, míg az APNG ugyan évek óta támogatott a legtöbb böngészőben, azonban a sokak által használt Internet Explorernek a legújabb, 10-es verziója is csak az első képkockát jeleníti meg.

A technikai akadályokon túl továbbra is probléma, hogy nem vezérelhető a megjelenítés. A képkockákat egyesével meg kell rajzolni, és össze kell őket fűzni. Innentől amikor a felhasználó amikor megnyitja az oldalt vagy képet, az végtelenített ciklusban fogja leját-

szani a képkockákat egymás után. Mivel raszteres állományról van szó a fájlméret viszonylag nagy lesz. Ezen kívül az elkészítése, módosítása nagyon problémás, időigényes.

2.4 Videó

Videó fájlban bármilyen animáció eltárolható, így ábrázolás szempontjából ezzel érhetőek el a leglátványosabb eredmények. [2.3. ábra]

2.3. ábra: Képkocka egy videóanimációból

Emellett azonban rengeteg hátránya van. A fájlmérete nagy, interaktivitása kimerül a megállíthatóság, és visszatekerhetőség funkcióknál. Megjeleníthetősége erősen függ a célesz-köztől, az arra feltelepített szoftverektől, kodekektől. Nem megjósolható, hogy a felhasználó böngészője helyben lejátssza-e, vagy más alkalmazásban nyitja-e meg, és annak milyen képességei lesznek. A videó fájlnak az elkészítése is hosszadalmas, a látványos összeállítás készítése is szakértelmet igényel. Nehéz megtalálni az egyensúlyt a méret és a minőség között.

2.5 Flash animáció

Az elmúlt évtized egyik legnépszerűbb animációs eszköze volt a weboldalakon. Eredeti célja vektorgrafikus animációk készítése volt, azonban ahogy fejlődött, egyre több funkci-óval vértezték föl. Az ActionScript programozási nyelvnek köszönhetően szinte bármi-lyen program megírható benne. Interaktív kezelőfelületet képes biztosítani a felhasználó-nak, képes lejátszani hangot, videót.

A platform hátránya, hogy szükséges hozzá egy Flash lejátszó. Ez sokáig természetesnek tűnhetett, hogy elérhető a felhasználók gépén, bár rossz hatásfoka már addig is észlelhető volt a nem Windows operációs rendszert futtató gépeken. Az okostelefonok és tabletek megjelenésével azonban előtérbe kerültek a hiányosságok. A környezet billentyűzetre és egérre lett kitalálva, így nem tudja megfelelően kezelni az érintőkijelzőket, illetve a kis képernyőméret sem kedvez neki. Kezdetben a fejlesztők próbálták kiküszöbölni a hibákat, ám végeredményben kijelenthető, hogy a mobileszközök, és a rajtuk futó böngészők gyenge vagy semmilyen támogatást nem nyújtanak Flash tartalmak lejátszására. (Winokur, 2011)

2.6 Egyéb beépülők

Animált térkép készíthető különböző böngészőbe épülő alkalmazásokkal is, mint például a Google Föld Plug-in. [2.4. ábra] Azonban a Flashhez hasonlóan ezeket is külön telepíteni kell a böngészőhöz, és ezek legtöbb esetben csak asztali gépekre, Windows alatt érhetőek el, esetleg Mac OS X-re, mint a Google Föld esetében is.

2.4. ábra: Animáció Google Föld Plug-in használatával

2.7 HTML5 canvas

A HTML5-ben megjelent a canvas elem, mely egy dinamikusan méretezhető és újrarajzolható rasztergrafikus felületet biztosít a böngészőben. A rajzolás JavaScript segítségével történhet rá.

Bár új elemet rárajzolni dinamikusan lehet, törölni csak az egész rajzot tudjuk. Ennek következtében ha csak egy elemet szeretnénk eltávolítani, az egész rajzot újra kell rajzoltatnunk a böngészővel, ami nem tekinthető optimális megjelenítésnek, cserébe viszont 3D-s megjelenítés esetén a támogatott eszközökön hardveres gyorsítás is elérhető.

Bár a HTML5 szabvány még nem végleges, magának a canvas elemnek a támogatottsága a böngészőkben általánosnak mondható. A ma használt asztali, illetve mobil böngészők közül kizárólag az Internet Explorer 8-as vagy korábbi verziói nem támogatják. Ugyan a még ma is igen elterjedt Windows XP operációs rendszerre ez a legutolsó verzió, azonban egy kiegészítő telepítésével ez a böngésző is képessé tehető a canvas megjelenítésére. (Paul, 2008)

3 Az alkalmazáshoz felhasznált technológiák ismertetése

3.1 PHP

A PHP programozási nyelv egy szerver oldali script nyelv. Tervezésekor csupán egy makrókészlet volt, mely honlapok karbantartásában segédkezett. Ám ezen hamar túlnőtt, és önálló programnyelvvé vált, mely dinamikus weboldalak készítésére használható, ennek megfelelően gyorsan és hatékonyan lehet vele HTML kódokat generálni, űrlapokat feldolgozni, és az elterjedt adatbázisszerverekhez is biztosít elérést.

A PHP előnye a kész függvényekben rejlik, melyeket arra optimalizáltak, hogy kényelmesen, gyorsan lehessen dinamikus oldalakat generálni különböző adatforrások, illetve a felhasználó visszajelzései alapján. Ennek köszönhetően gyorsan lehet benne fejleszteni, és a rá bízott feladatokat nagyon hatékonyan tudja elvégezni.

3.2 SVG

Az SVG (Scalable Vector Graphics) egy XML alapú leíró nyelv, amivel vektoros grafikát lehet meghatározni. Ez használható vektorgrafikus rajzprogram adatfájljaként (pl.: Inkscape), de a modern böngészők is képesek megjeleníteni az ily módon leírt grafikát beágyazva a HTML kódba. Ennek előnye, hogy nem csak egy előre megrajzolt képet lehet vele megjeleníteni, hanem akár valós időben is szerkeszthető felhasználói oldalról JavaScript segítségével.

Összehasonlítva a fentebb már bemutatott canvas elemmel, az SVGnek megvan az az előnye, hogy az objektumok kirajzolás után is önállóak maradnak, így nem csak megjeleníteni, de eltüntetni is lehet őket anélkül, hogy az egész rajzterületet újra kéne rajzoltatni. Ez jóval kényelmesebben kezelhetővé, és jóval erőforrás-kímélőbbé teszi ezt a technológiát.

Továbbá az SVG külső fájlok beemelését is lehetővé teszi, így akár egy raszter képet is megjeleníthetünk a vektoros objektumok között. Mivel nem egy elemről van szó, amiben csak képpontok találhatóak, mint a canvas esetében, az SVG-ben kirajzolt pontok, vonalak, felületek interaktívak is lehetnek: ha rámutatunk az egérrel, vagy rákattintunk egy

elemre, az végrehajthat JavaScript utasításokat is, így akár egész kezelőfelületet is írhatunk a segítségével.

Az SVG megjelenítési kompatibilitása a böngészőkben megegyezik a canvas tulajdonságaival, tehát csak az Internet Explorer 8-as vagy régebbi verziói nem támogatják, ám egy plugin segítségével ez a probléma is kiküszöbölhető. (Savarese, 2012)

3.3 JavaScript

A PHP-val ellentétben a JavaScript olyan script nyelv, ami a felhasználó böngészőjében fut. Ennek köszönhetően nem kell az adatokat visszaküldenünk a szervernek ahhoz, hogy reagálhassunk a felhasználó beavatkozásaira. Szintén rengeteg előre megírt függvénnyel rendelkezik, melyek segítségével valós időben módosítható a HTML oldal tartalma, és akár a HTML-be ágyazott SVG tartalom is, mint azt majd láthatjuk a későbbiekben.

4 A megvalósított alkalmazás rövid ismertetője

4.1 Adatok

A weboldal a térképi adatokat Google KML állományból olvassa be. A felhasználó feltöltheti saját adatfájlját, vagy választhat a szerveren lévők közül egyet.

4.2 Feldolgozás

A szerveren a PHP program beolvassa a térbeli és időadatokat a KML fájlból.

A WGS84 ellipszoidi koordinátákból vetületi koordinátákat számol. Ebből kiszámítja a szélső koordinátákat, és a weboldalon megjelenő kép méretének függvényében a szükséges nagyítási értéket. Elkészíti a HTML kódba ágyazandó vektorgrafikus (SVG) kódot külön-külön a pontszerű, vonalas és felületi elemekre, és hivatkozást generál az animáció hátteréül szolgáló statikus térképhez.

Az időadatokból meghatározza az animáció kezdő- és végidőpontját. Ha a felhasználó nem adott meg lépésközt, akkor meghatározza, hogy egy egységnyi idő alatt mennyi valós idő teljen el az animációban. Ezen adatok alapján kiszámítja, hogy az egyes rajzi elemek mikor jelenjenek meg, illetve tűnjenek el.

4.3 Megjelenítés

A böngésző a megkapott hivatkozás alapján megjeleníti az OpenStreetMap szerveréről letöltött háttértérképet. Az animáció elindításakor a JavaScript kód sorra veszi a térképi elemeket, és a megfelelő időpontban megjeleníti, illetve eltünteti őket a képernyőről.

Eközben a térkép jobb oldalán egy idősávban látható az animáció kezdő- és végidőpontja, illetve hogy a kettő között éppen hol tartunk.

5 A feladat megvalósítása

5.1 Az induló oldal (index.php)

Mikor a felhasználó belép az oldalra az `index.php` által generált tartalom fogadja. Ennek egy része statikus HTML kód, míg más kódrészeket a PHP állít elő.

A fájl első sora a a DOCTYPE-ot tartalmazza, mely a böngésző számára azonosítja a weboldal típusát. Ez megmondja, hogy a további tartalmat az XHTML 1.0 Transitional szabvány szerint kell értelmezni.

5.1.1 Az induló oldal PHP kódrésze

A következő részben a PHP programkódokat ismertetem. Ezeket a kódokat a böngésző nem kapja meg, hanem a PHP értelmezi még szerver oldalon.

Az alkalmazás a `session_start()` függvény meghívásával létrehoz egy egyedi azonosítót és egy változót, melybe az adott felhasználó adatai kerülnek (pl. kiválasztott fájl azonosítója, idő adatok).

A jobb átláthatóság és a biztonság növelése érdekében a különböző feladatokat ellátó programkódok külön függvénybe kerültek, és az ezeket tartalmazó fájlok nem érhetőek el közvetlenül az internetről. Ezek a fájlok egy külön munkakönyvtárban találhatóak, melynek elérési útját a `globals.inc.php` tartalmazza. Ennek az is az előnye, hogy ha másik szerverre kéne költöztetni az alkalmazást, akkor az elérési utakat csak ebben a `globals.inc.php` fájlban kéne átírni.

A következő parancs tehát ennek a fájlnak a betöltése, majd az ebben található állandók felhasználásával a munka könyvtáron belül található `svg_anim.inc.php` és a `php_google_maps/staticmap.inc.php` fájlok betöltése, melyek a további szerveroldalon használt függvényeket tartalmazzák.

A weboldalon megjelenő térkép méretét a `$width` és a `$height` változók tartalmazzák. Ezek értékét beállítottam 800 és 500 képpontra, ami a manapság használt kijelzőkön egy jól látható nagy képet, és körülötte egy kényelmesen elférő weboldalt eredményez. A kézi eszköz egy részének, főleg mobiltelefonoknak ugyan kisebb a felbontása, mint az egész weboldal mérete, azonban ezeknél az eszközöknél a böngésző alapértelmezés szerint

akkorára kicsinyíti az oldalt, hogy az beférjen a képernyőbe, és a felhasználó könnyedén tud egyes részletekbe belenagyítani, arrébb görgetni az oldalt.

A `manageSession()` függvény meghívása biztosítja, hogy a felhasználó kérései teljesüljenek. Ez kezeli a kiválasztott térkép, és az animáció sebességének tárolását.

A `listKmlFiles($fileid)` függvény elkészíti a szerveren elérhető KML fájlok listáját, és kiválasztja közülük az aktuálisan megjeleníteni kívánt állományt.

A `readKml($fname)` függvény végzi a kiválasztott állományból az adatok beolvasását a későbbiekben felhasználható PHP változókba.

A `calcTime($datas, $timeval)` függvény feladata az adatokból meghatározni, hogy melyik a legkorábbi és a legkésőbbi időpont az animációban, és hogy az animáció sebességét ennek alapján meghatározza.

A `calcDataExtents($datas)` függvény a geometriai adatokból kiszámítja a szélső koordinátaértékeket. Ezt az adatot rögtön tovább is adja a `calcImgExtents($dataextents, $width, $height)` függvénynek, ami a képméret és a szélső adatok alapján meghatározza, hogy mik is lesznek a térkép sarok- és középponti koordinátái, és hogy melyik az optimális nagyítási szint.

A `calcProjdatas($latmax, $lonmax, $latmin, $lonmin)` függvény a földrajzi sarokkoordinátákból számít vetületi koordinátákat, illetve meghatározza a vektorgrafikus állomány kép vetületi méreteit.

5.1.2 Statikus HTML kód

Ebben a szekcióban a weboldal fizikai szerkezete kerül meghatározásra. A HTML kódba helyenként PHP hívások is megjelennek, melyek biztosítják a képméret meghatározását, és az ehhez illeszkedő egyéb megjelenő elemeket, mint például a térkép ismertetőjét, vagy az egyes térképi objektumok leírását. Továbbá ebben a szekcióban kerül meghívásra a `writesvg($polyarray, $linearray, $pointarray, $timedata, $sortbegin, $sortend, $height, $svgheight)` függvény is, ami elvégzi a vektorgrafikus adatok SVG formátumba öntését is, hogy a böngésző képes legyen a megjelenítésükre.

Először is meg kell adni a böngészőnek, hogy HTML kód következik. Ezt követően a fejlécben (<head>) megtörténik a karakterkódolás beállítása, a weboldal címének megadás, az esztétikus megjelenésért felelő CSS stíluslap elérésének beállítása, és az animáció működéséhez szükséges JavaScript kódokat tartalmazó fájlok betöltése.

A felhasználó számára a weboldal látható részét a törzs (<body>) rész foglalja magába.

Az ezen belül található első doboz ( ) az oldal címét tartalmazza, majd egy figyelmeztetést, ami akkor jelenik meg, ha a böngésző nem alkalmas a JavaScript futtatására, végül egy leírás rövid és hosszabb változatát az oldalra látogató felhasználó számára. Ha a felhasználó Windows operációs rendszert és a Firefox böngésző 17-nél újabb változatát használja, akkor megjelenik még egy figyelmeztetés, illetve leírás, ami ezen böngésző megjelenítési hibájáról, és a hiba megoldásáról tájékoztatja.

A következő dobozon belül helyezkedik el az összes többi tartalom.

A felhasználó által beállítható tartalmak két dobozba (és ezzel együtt két sorba) kerültek. Felül a térkép kiválasztását biztosító lista, és a saját térkép feltöltését lehetővé tevő elem található, alatta pedig az animáció indítása, leállítása gombok, illetve a sebesség beállítását biztosító eszköz található. A térkép kiválasztás, feltöltés és a sebesség beállítás <form> elemben helyezkedik el, ami lehetővé teszi, hogy a felhasználó visszaküldhesse a beállításokat a szervernek, mely az oldal újratöltésével érvényesíti azokat.

Ha a felhasználó által feltöltött fájlt nem sikerül feldolgozni, akkor erről a következő dobozban kap értesítést. Ha nincs hiba, nem jelenik meg értesítés.

A térkép fölött még egy doboz jelenik meg, ami az aktuálisan kiválasztott térkép leírását, illetve a térképről egy kiválasztott elem leírását tartalmazza, ha ezen adatok elérhetőek. A térkép leírását a PHP helyezi el az oldalon, az elem leírását viszont már JavaScript attól függően, hogy a felhasználó éppen melyik térképi elemre nyomott rá.

Maga a térkép is külön dobozban található, melynek szélességét a PHP állítja be. Ezen belül helyezkedik el a későbbiekben tárgyalt SVG kód.

A térkép jobb oldalán egy újabb dobozban jelennek meg az időadatok. A doboz aljába és tetejébe a PHP írja be a kezdő és vég időpontokat, illetve beállítja a doboz méretét, hogy illeszkedjen a térkép méretéhez. Az idősáv növekedéséről már a JavaScript gondoskodik.

A statikus részben csak az ehhez szükséges dobozok vannak definiálva, hogy aztán ezeket manipulálhassák a megfelelő JavaScript függvények.

5.1.3 SVG kód

Mint már fentebb említettem, az SVG előnye, hogy tartalma közvetlenül beleírható a HTML-be, mintha maga is csak HTML kód lenne: az `<svg> </svg>` kulcsszavak közé írható be a vektorgrafika kódja maga.

Mivel az SVG szabadon méretezhető vektorgrafikus képtípus, így mindenképpen meg kell adni egy méretet, különben a megjelenítő nem fogja tudni, hogy a beírt koordináták hány képpontnak felelnek meg. Ezért az `<svg>` elemen belül meg kell adni a szélességet és a magasságot a `width` és `height` tulajdonságokkal. Alkalmazásomban ezt is a PHP végzi el.

További különlegessége az SVG-nek, hogy nem kell az egész képet megjeleníteni a 0,0 koordinátától a maximális x és y koordinátáig, hanem egy nézetet (`viewBox`) lehet benne definiálni, ami a határkoordinátáknál akár kisebb, akár nagyobb lehet. A `viewBox`nak négy paramétert kell megadni, a kezdő koordinátákat, amik lehetnek negatívak is, a szélességet és a magasságot. Az így megadott területet fogja belekicsinyíteni, vagy nagyítani a képpontokban megadott képméretbe, akár az oldalarányok megsértésével is (hacsak ezt meg nem tiltjuk neki). Ezáltal lehet megadni, hogy csak az animáció szempontjából hasznos területet ábrázoljuk, és azt is jól illesszük rá a háttértérképre.

A háttértérkép beemelése egy `<image>` elem segítségével könnyedén megvalósítható. A PHP kitölti a már rendelkezésre álló adatokat, a kép – elhelyezéséhez szükséges vetületi koordináta-rendszerben vett – sarokkoordinátáit, szélességét, hosszúságát, illetve az OpenStreetMap szerverére mutató linkbe az ábrázolni kívánt terület középpontjának földrajzi koordinátáit, a nagyítás szintjét és a képpontokban vett képméretet.

Például: `http://ojw.dev.openstreetmap.org/StaticMap/?lat=46.950858845&lon=20.37563883&z=6&w=800&h=500&att=none&show=1`
Látható, hogy egy `StaticMap`, tehát statikus térkép kerül meghívásra, aminek a középponti koordinátái φ=46,950858845°, λ=20,37563883°, a nagyítási szint 6, a képméret 800×500 képpont. Ezen kívül még meg kell adni, hogy a megjelenjen-e a szabad felhasz-

nálást mutató CC-BY-SA 2.0 szerzői jogi pecsét, illetve hogy csak a kép jelenjen meg, ne pedig az azt generáló weboldal.

Általános probléma az informatika, és a földrajzi ábrázolás keveredésénél, hogy míg a földrajzi koordináta-rendszerek északkeleti tájékozásúak – tehát a képernyőn a koordinátáknak jobbra és fölfele kéne nőniük –, addig a nem térképészeti céllal készült alkalmazásokban az elfogadott irány a lefelé és jobbra növekvés. Tehát míg a vetületi X koordinátával nincs problémánk, addig az Y épp ellenkező irányba nő, mint ahogy mi szeretnénk, és az origója sem a bal alsó, hanem a bal felső sarokban van.

Nincs ez másként az SVG esetében sem, azonban lehetőség van a koordináta-rendszer eltolására és megfordítására is. Ehhez a `transform` parancsot kell használni, melynek `translate(0, y_eltolas)` tulajdonsága eltolja az y tengely kezdőértékét a PHP által kiszámolt vetületi értékkel, míg a `scale(1, -1)` tulajdonsága megfordítja az Y tengely irányát. (W3C, 2011)

Ha mindezek megvannak, meghívható a `writesvg($polyarray, $linearray, $pointarray, $timedata, $sortbegin, $sortend, $height, $svgheight)` PHP függvény, ami feltölti – egyelőre láthatatlan – tartalommal az SVG fájlt. A megjelenítésről majd a megfelelő JavaScript függvény gondoskodik az animáció elindítását követően.

5.2 A Google KML fájlokat kezelő PHP függvények

5.2.1 listKmlFiles($fileid)

Ennek a függvénynek két feladata van. Egyrészt elkészíti a szerveren található KML fájlok listáját, melyből a felhasználó választhat, másrészt a paraméterként kapott azonosító alapján megadja, hogy melyik fájlt kell éppen feldolgozni. Azért van szükség az azonosítóra, hogy a felhasználó ne tudjon közvetlenül fájl elérési utat küldeni a szervernek, így nem olvastathatja be a programmal bármilyen fájl tartalmát, csak a listában szereplőekét.

Visszatérési értéke egy asszociatív tömb, melynek két eleme van. A KML fájlokat a munkakönyvtáron belüli `sourcedata` mappában keresi, és ebből állítja össze a fájllistát,

ami kikerül a weboldal legördülő menüjébe (`filelist`). Ezen kívül megadja, hogy a `$fileid`hoz tartozó fájlnak mi az elérési útja (`file`).

5.2.2 readKml($fname)

Ahogy a nevéből is látható, ez a függvény az adatok beolvasását végzi el a kiválasztott Google KML fájlból. Paraméterként csak a fájl nevét kapja meg, amiből maga állítja elő a teljes elérési utat.

Ha a felhasználó töltött fel saját fájlt, akkor azt a `$_FILES` tömbben éri el a PHP. Ennek tartalmát ellenőrizve eldönthető, hogy sikeres volt-e a feltöltés, vagy valami hiba adódott (pl.: túl nagy fájl, megszakadt a feltöltés, nem sikerült a szerveren eltárolni a fájlt). Ha ezen az ellenőrzésen sikeresen átjutott a függvény, akkor a felhasználó fájlját próbálja feldolgozni. Ha hibás a feltöltés vagy nem történt feltöltés, akkor a paraméterként kapott fájlnévből dolgozik tovább.

A PHP-ban két egyszerűen használható osztály is van, amibe be lehet olvasni XML fájlok tartalmát: a `DOMDocument` és a `SimpleXML`. Mivel a Google KML állomány egy szabványos XML fájl, ezért ezt a két eszközt nagy biztonsággal lehet használni a fájl tartalmának kezeléséhez, és nem szükséges azt szövegként értelmezve manuálisan feldolgozni.

A `DOMDocument`ben a `getElementsByTagName` metódussal meg lehet találni az összes adott azonosítójú elemet. Ezzel az eszközzel nyeri ki a függvényem az állományból a dokumentum leírását (`description`), a helyszíneket (`Placemark`), és dönti el, hogy éppen pont (`Point`), vonal (`LineString`) vagy poligon (`Polygon`) elemről van-e szó. A `getElementsByTagName` visszatérési értéke egy tömb, aminek az összes elemén végig lehet haladni a `foreach` PHP paranccsal, így kinyerve egyesével például a helyszíneket.

A helyszínekből a következőként tárgyalt `place2Arr($place, $i)` függvényem nyeri ki az adatokat.

A `DOM` elemeken könnyű navigálni, azonban nehézkes kinyerni belőlük az adatot, ezért érdemes a `simplexml_import_dom` paranccsal `SimpleXML` elemet csinálni belőlük, melynek a visszatérési értéke már maga az adat, ha csak egy elem van benne. Azonban ha szóközök, tabulátorok, új sorok vannak a keresett szöveg előtt, mögött, akkor azok bekerülnek a kinyert szövegbe is, ezért szükséges a `trim` parancs használata, mely eltá-

volítja ezeket. Ha a végeredményt ki akarjuk íratni közvetlenül a HTMLbe, akkor még a `htmlspecialchars` függvényt is le kell futtatni, mely elkódolja a HTML vezérlőkaraktereket.

A függvény visszatérési értéke egy asszociatív tömb, mely tartalmazza a térkép leírását (`mapdescription`), és három tömböt a pont (`pointarray`), vonal (`linearray`) és poligon (`polyarray`) elemekkel.

5.2.3 place2Arr($place, $i)

A Google KML állományban egy `Placemark` elem több geometriai objektumot is tartalmazhat, ha definiálva van benne a `MultiGeometry` elem. Ez akkor fordul elő például, ha egy nemzeti park több poligonból áll. Ilyen esetekben ez a függvény többször is megkapja az adott elemet a `$place` változóban, és a `$i` változó adja meg neki, hogy hányadik geometriát kell éppen kiolvasnia. Mivel az SVG nem támogatja az ilyen `MultiGeometry` elemeket, csak ha jóval bonyolultabbá teszem a szerkezetet, ezért inkább külön elemekre bontottam őket, melynek minden tulajdonsága megegyezik az azonosítóján és a geometriáján kívül.

A függvény visszatérési értéke egy tömb, mely tartalmazza az adott elem nevét, megjelenési és eltűnési időpontját és geometriai adatait, egyelőre csak szövegként.

5.3 A geometriát feldolgozó PHP függvények

5.3.1 calcDataExtents($datas)

A `$datas` változóban megkapja az összes rajzi elemet, amiből megkeresi a szélső koordinátákat. Visszatérési értéke egy asszociatív tömb, mely a Google KMLből vett elemek minimális (`dlatmin`, `dlonmin`) és maximális (`dlatmax`, `dlonmax`) földrajzi koordinátáit tartalmazza.

5.3.2 calcImgExtents($dataextents, $width, $height)

Ez a függvény megkapja az előző függvény által kiszámított szélső értékeket, és a megjelenítendő kép szélességét és magasságát. Ebből számítja ki, hogy mik lesznek a végleges

kép minimális, maximális, középponti földrajzi koordinátái, és hogy hányas nagyítási szinttel kell megjeleníteni a tartalmat, hogy elférjen a képen.

Ahhoz hogy kiszámíttassam a fenti adatokat, egy külső függvénykönyvtárat használtam fel. A Mika Tuupola által fejlesztett `PHP Google Maps` egy olyan PHP függvénykönyvtár, melyet statikus Google Maps térképek kezelésére készített a fejlesztője. Mivel az OpenStreetMap vetülete megegyezik a Google Maps vetületével, az eredmény számomra is megfelelő. (Tuupola, 2008)

Az alkalmazás meg is tudná jeleníteni a megadott paraméterek alapján a statikus térképet, nekem azonban csak az adatokra volt szükségem. A Google Maps hátránya egyébként, hogy legnagyobb képméret, amit elő lehet vele állítani statikus térképből a 640×640 képpont.

A függvény létrehoz egy statikus térképobjektumot (`Google_Maps::create('static')`), megadja a méreteit képpontokban (`setSize`) és hozzáadja a minimális és maximális koordinátákat (`Google_Maps_Coordinate`) jelölőelemként (`Google_Maps_Marker`). Ezt követően tudja megállapítani a nagyítási szintet, amibe belefférnek a szélső értékek (`zoomToFit`), és megadni az adott nagyítási szinten a kép végleges szélső értékeit (`getBounds`) még mindig földrajzi koordinátákkal.

A függvényem visszatérési értéke egy asszociatív tömb, mely tartalmazza a minimális és maximális hosszúsági és szélességi koordinátákat (`lonmin`, `latmin`, `lonmax`, `latmax`), a kép középponti koordinátáját (`loncenter`, `latcenter`) és a nagyítási szintet (`zoomlevel`).

5.3.3 calcProjdatas($latmax, $lonmax, $latmin, $lonmin)

Ez a függvény a minimális és maximális földrajzi koordinátákból kiszámítja a a vetületi koordinátákat, az SVG állomány koordinátákban vett magasságát és szélességét, és hogy mennyivel kell megjelenítéskor eltolni a koordináta-rendszert, és ezeket visszaadja egy asszociatív tömbben (`minx`, `miny`, `maxx`, `maxy`, `svgwidth`, `svgheight`, `svgtranslate`).

5.3.4 pointLonLat2Xy($llcoords)

Egy pont elem földrajzi koordinátáit alakítja át vetületi koordinátává a `lon2x($lon)` és a `lat2y($lat)` függvények segítségével. A koordinátákat egy tömbben adja vissza.

5.3.5 lonLat2Xy($llcoords)

Egy vonal vagy poligon elem földrajzi koordinátáinak sorát alakítja át vetületi koordinátákká a `lon2x($lon)` és a `lat2y($lat)` függvények segítségével. A koordinátalistát egy szövegváltozóban adja vissza x1 y1 x2 y2 … formában, ahogy aztán az SVG-be kell kerülnie.

5.3.6 lon2x($lon)

WGS84 ellipszoidi hosszúságot gömbi mercator (EPSG:3857) vetületi x koordinátává alakít az $x = R * \lambda$ képletnek megfelelően.

5.3.7 lat2y($lat)

WGS84 ellipszoidi szélességet gömbi mercator (EPSG:3857) vetületi y koordinátává alakít az $y = \ln\left(\tan\left(\frac{\Pi}{4} + \frac{\varphi}{2}\right)\right) * R$ képletnek megfelelően.

Ennek a két képletnek a különböző programnyelveken vett implementációi elérhetőek az OpenStreetMap Wiki oldalán. (OpenStreetMap Wiki, 2013)

5.4 Az időadatokat feldolgozó PHP függvények

5.4.1 calcTime($datas, $timeval)

Ez a függvény végigmegy az összes ábrázolt elemen és kiolvassa belőlük az induló és vég időadatokat. Mindkét adatsort növekvő sorba rendezi, majd meghatározza a legkorábbi és a legkésőbbi időpontot. Kiszámítja a két időpont különbségét, ezzel meghatározva az animáció valós időben vett hosszát.

A lépésközt, hogy mennyi idő teljen el két animációs lépés között, a felhasználó is megadhatja. Ha ez nem történt meg, akkor a függvény egy egyszerű feltétel alapján választ. Ennek ismeretében végül kiszámítja, hogy hány lépésből fog állni az animáció.

Visszatérési értéke ismét egy asszociatív tömb, mely tartalmazza a sorba rendezett kezdőidőpontok (`sortbegin`) és végidőpontok (`sortend`) tömbjét és még egy asszociatív tömböt (`timedata`), ami tartalmazza a legkisebb (`mintime`) és legnagyobb (`maxtime`) időt, a lépésközt (`timeval`) és a lépések számát (`timesteps`).

5.4.2 timeOptions($timetype)

Ez a függvény készíti el a legördülő listát, amiből a felhasználó kiválaszthatja, hogy az animációs lépésköz mekkora legyen. Paraméterként megkapja, hogy mi az aktuális lépésköz, így a listában az jelenik meg kiválasztva.

5.4.3 printTime($time, $timeval)

A térkép melletti idősáv kezdő és végidőpontjának kiírásához készíti el a szöveget ez a függvény. Ha a lépésköz kisebb egy napnál, akkor óra, perc, másodperc értékeket is kiírat (Y. m. d. H:i:s formában), különben csak dátumot (Y. m. d. formában).

5.5 Egyéb PHP függvények

5.5.1 manageSession()

Ennek a függvénynek a feladata, hogy a felhasználó által adott utasításokat eltárolja a `$_SESSION` globális tömb változóba.

Ha a felhasználó kiválaszt a szerveren lévő Google KML fájlok közül egyet, akkor annak az azonosítója tárolásra kerül a `$_SESSION['fileid']` változóba, egyébként a 0 azonosítójú fájl lesz kiválasztva.

Ha a felhasználó beállítja a lépésközt az aktuális animációhoz, akkor az is eltárolásra kerül a `$_SESSION['timenum']` és `$_SESSION['timetype']` változókba, egyébként a változó üresen marad, és a `calcTime($datas, $timeval)` függvény fogja kiszámítani az optimális időközt.

5.5.2 writesvg($polyarray, $linearray, $pointarray, $timedata, $sortbegin, $sortend, $height, $svgheight)

Ez a függvény írja ki az `index.php`ba az SVG elemeket, és számítja ki a hozzájuk tartozó időadatokat, mely alapján a JavaScript függvények megjelenítik, illetve eltüntetik őket.

A rajzi elemek méretét dinamikusan határozza meg a függvény az SVG vetületi koordinátákban vett magasságának és a kép képpontban mért magasságának hányadosa alapján. Így lesz például a pont elemek sugara 4 képpont függetlenül a nagyítási szinttől.

A függvény végig megy az összes rajzi elemen, és attól függően, hogy pont, vonal, vagy felület elemről van-e szó, először átalakítja a koordinátáikat a `pointLonLat2Xy($llcoords)` vagy a `lonLat2Xy($llcoords)` függvény meghívásával, majd kiírja a megfelelő tartalmat a böngésző számára az SVG formátumában: `circle` (pontok), `polyline` (vonalak) vagy `polygon` (poligon).

Az elemek azonosítója (`id`) típust és sorszámot tartalmaz, ami alapján el tudja végezni a JavaScript a ki-be kapcsolást.

A pontnak x (`cx`) és y (`cy`) koordinátája van, míg a vonalnak és a felületnek koordinátalistája (`points`).

A pontnak meg kell adni a sugarát (`r`) és a kitöltési színét (`fill`), a vonalnak a vastagságát (`stroke-width`) és a színét (`stroke`), míg a poligonnak csak a kitöltési színét (`fill`).

A következő beállítások minden elemre vonatkoznak. Van egy átlátszósági értékük (`opacity="0.7"`), hogy ne takarják ki a háttértérképet. Van nevük (`name`), ha elérhető volt a Google KML fájlban, hogy az elemre kattintva az megjelenhessen az oldalon. Továbbá beállítottam, hogy az elemek ne jelenjenek meg maguktól (`style="display: none"`), csak ha a `show()` JavaScript függvény megjelenítteti őket, és egy kattintási eseményt (`onClick`), ami meghívja a `showItemDesc(item)` JavaScript függvényt.

Az elemek megjelenítéséhez és eltüntetéséhez szükséges időadatokat úgy számítja ki a függvény, hogy az elemekhez tartozó időből kivonja a kezdőidőt, és a különbséget elosztja a megadott lépésközzel. Így a JavaScript már animációs időegységben kapja meg,

hogy mit mikor kell megjeleníteni vagy eltüntetni. Az eredmények a `$_SESSION['animdata']` tömbbe kerülnek, ahonnan a következő pontban ismertetett `anim.js.php` alakítja át a JavaScript számár értelmezhető formátumba.

5.5.3 anim.js.php

Megvalósítását tekintve ez nem függvény, hanem fájl, ami dinamikusan állítja elő az animációhoz szükséges időadatokat tartalmazó JavaScript változókat a PHP adattömbjéből. A PHP tömbből JavaScript tömbbé történő átalakításhoz a `my_json_encode($arr)` függvényt használja.

5.5.4 my_json_encode($arr)

Ezt a függvényt a PHP hivatalos oldaláról vettem át. Feladata, hogy a PHP tömbből JavaScript tömböt készítsen. Erre létezik a PHPban egy előre megírt `json_encode` függvény, azonban az egyetemi szerveren még régebbi verziójú PHP található, ami nem támogatja, hogy az UTF-8 kódolású karakterek a JavaScript tömbben is UTF-8 kódolásban maradjanak. Ezeket a karaktereket \uXXXX formában elkódolná, amit aztán JavaScriptben kéne visszaalakítanom. Ezt a bonyodalmat megkerülendő használtam az egyik felhasználó által írt függvényt, ami rendelkezik ezzel a képességgel is. (PHP, 2011)

5.6 Az animációért felelős JavaScript függvények

Ezek a függvények az `animsvg.js` fájlban találhatóak, melyet az `index.php` tölt be. A fájl tartalmaz két, függvényeken kívüli, globális változót. A `timed` tömb tartalmazza a már beidőzített elemek animációs lépéseit, hogy az animáció megállításakor törölni lehessen őket. A `timelinea` tömb pedig az idővonal animációs lépéseit ugyanezen célból.

5.6.1 show()

Ez a függvény végigmegy az `animdata` tömb összes elemén. Ha az adott elem kezdőideje nincs beállítva (`begin == null`), akkor azt rögtön megjeleníti, egyébként meghívja a `doShow(id, time)` függvényt.

5.6.2 doShow(id, time)

Paraméterként megkapja, hogy melyik elemet (`id`), és mennyi animációs idő múlva (`time`) kell megjelenítenie. Ennek alapján beállít egy időzítőt (`setTimeout`), ami elvégzi a megjelenítést. Az időzítő azonosítóját pedig elmenti a `timed` tömbbe.

5.6.3 hide()

Ez a függvény végigmegy az `animdata` tömb összes elemén. Ha az adott elem végideje be van állítva (`end !== null`), akkor meghívja a `doHide(id, time)` függvényt.

5.6.4 doHide(id, time)

Paraméterként megkapja, hogy melyik elemet (`id`), és mennyi animációs idő múlva (`time`) kell elrejtenie. Ennek alapján beállít egy időzítőt (`setTimeout`), ami elvégzi az elrejtést. Az időzítő azonosítóját pedig elmenti a `timed` tömbbe.

5.6.5 hideall()

Ez a függvény végzi az animáció leállítását, ha a felhasználó rányom a `Törlés` gombra vagy újraindítja az animációt.

Először végigmegy a `timed` és a `timelinea` tömbök összes elemén, és törli az összes időzítőt, majd a tömbök tartalmát is. Ezt követően az `animdata` tömbön is végigmegy és az összes rajzi elem megjelenítését kikapcsolja. Végül alaphelyzetbe állítja az idősávot.

5.6.6 timeline(linesteps, mintime, maxtime)

Ennek a függvénynek a feladata, hogy kiszámítsa, hogy melyik animációs lépésben mekkorának kell lennie az idővonalnak, és hogy éppen milyen dátumot kell mellé kiírni.

Paraméterként megkapja a lépések számát, a minimális és a maximális időpontot. A HTML kódból kinyeri, hogy mekkora hely áll rendelkezésre az idővonalnak `timediv` dobozon belül a `timebegin` doboz alatt és a `timeend` doboz fölött. Ezt a magasságot elosztja a lépések számával, így megkapva, hogy egy lépésben hány képponttal kell nőnie a vonalnak.

Az idővonalnak a kitöltése színátmenetes, ezért a mérete nem változhat, csak az őt befoglaló doboz változik, így az abból kilógó vonalrész nem látszódik.

A függvény a kezdő és végdátum különbségéből és a lépések számából minden lépéshez meghatározza az idővonal mellé kiírt szöveget is.

Ezt követően minden lépéshez meghívja a `sizeTimeline(i, h, text)` függvényt ami az idővonal animálásáért felel.

5.6.7 sizeTimeline(i, h, text)

Ez a függvény a paraméterként kapott lépésszám (`i`), vonalméret (`h`), és szöveg (`text`) alapján beállítja az időzítőt, hogy mennyi idő múlva mekkorának kell lennie a vonalnak, és mit kell mellé írni. Az időzítő azonosítóját elmenti a `timelinea` tömbbe, hogy törléskor meg lehessen szakítani az animálást.

5.7 Egyéb JavaScript függvények

Ezek a függvények a `page.js` fájlban találhatóak, melyet az `index.php` tölt be.

5.7.1 checkFirefox()

A Firefox 17-nél újabb verziói esetén van egy megjelenítési hiba. Hogy erről tájékoztathassam a felhasználót ez a függvény ellenőrzi, hogy a hiba által érintett böngészőt használ-e.

A böngésző által tárolt userAgent azonosítóból kiolvassa, hogy Firefox-e a böngésző, és hogy verziószáma nagyobb-e 17-nél. Ha igen, akkor megjeleníti a leírást az oldalon, ami egyébként rejtve marad.

5.7.2 showFfError()

A rövid hibatájékoztatáson túl elő lehet hívni egy hosszabb értesítést is, ennek a függvénynek a dolga, hogy ezt megjelenítse.

5.7.3 showLongDesc()

Az oldal tetején először csak egy rövid ismertető jelenik meg az alkalmazásról, aminek a végén található egy link, arra kattintva fut le ez a függvény, ami megjeleníti a hosszabb leírást.

5.7.4 hideLongDesc()

Ezt a függvényt meghívva lehet eltüntetni a hosszabb leírást.

5.7.5 checkExt(file)

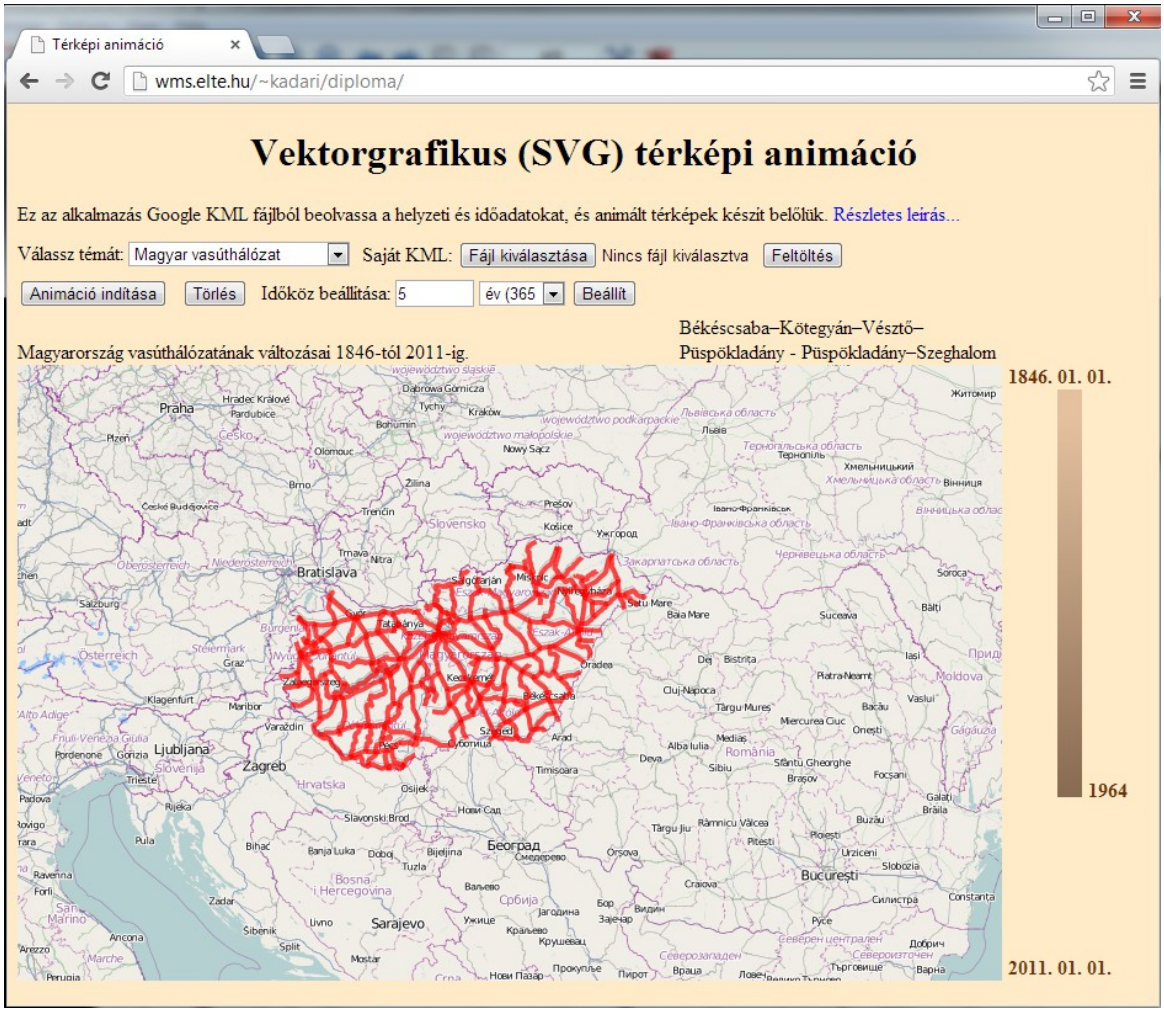
Amikor a felhasználó fel akar tölteni egy fájlt a szerverre, ez a függvény ellenőrzi, hogy valóban `.kml`-e a kiterjesztése. Ha nem, akkor megjelenít egy hibaüzenetet, és nem végzi el a feltöltést.

5.7.6 showItemDesc(item)

Ha a térképen egy elemre kattintunk, ez a függvény fut le. A paraméterként kapott térképi elemből kiolvassa a nevét (`name`), és tartalmát beleírja a térkép fölött található `itemdesc` azonosítójú dobozba.

6 Az alkalmazás használatának bemutatása

6.1 Az oldal szerkezete



6.1. ábra: Az oldal szerkezete

Az oldal nagy részét a háttértérkép foglalja el, melyet az OpenStreetMap szerveréről tölt be az alkalmazás. Erre kerül rá a céltematika animációja, tehát az idő előrehaladtával pontok, vonalak és felületek jelennek meg rajta, illetve tűnnek el róla. [6.1. ábra]

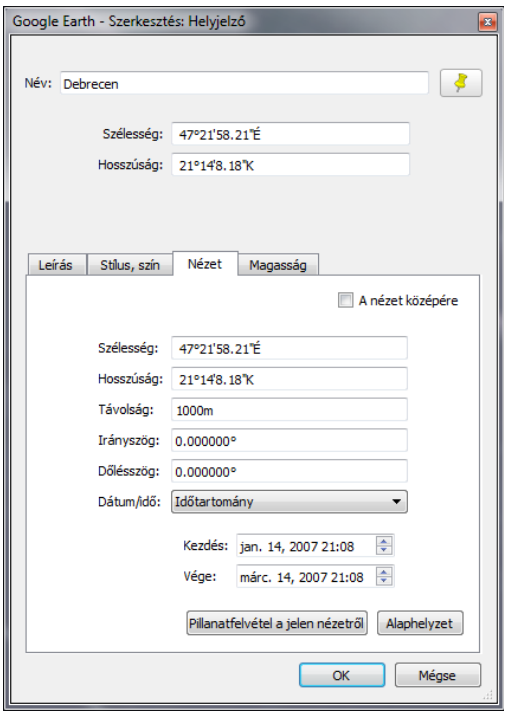
A térkép jobb oldalán található az idősáv, mely mutatja az animáció haladását. Ennek a tetején és alján látható a kezdő, illetve végidőpont, mellette pedig az aktuális idő.

A térkép fölött a bal oldalon az aktuális animáció leírása, jobb oldalon, pedig egy kattintással kiválasztott térképi elem leírása jelenik meg.

Az oldal tetején található a cím, az alkalmazás leírása, és a kezelőfelület.

6.2 Google KML fájl előkészítése

A Google KML fájlban az elemeknek kell, hogy legyen időbélyegzőjük (`TimeStamp`), vagy időtartományuk (`TimeSpan`). Ennek hiányában az elem folyamatosan látszódni fog az animáció alatt. Ha csak időbélyegzője van, az a megjelenési időt fogja megadni (`when`), míg az időtartomány a megjelenési (`begin`) és eltűnési (`end`) időt is definiálja. Ezeket az adatokat a Google Föld alkalmazásban az elem szerkesztésével, vagy új elem létrehozásakor a `Szerkesztés` ablakban a `Nézet` fülön adhatjuk meg. [6.2. ábra]



6.2. ábra: Időadatok szerkesztése a Google Földben

Ha más programmal hozzuk létre a KML fájlt, és az nem írja be az időadatokat, akkor akár egy szövegszerkesztővel is megtehetjük a szükséges kiegészítést. Annyi a dolgunk, hogy a `<Placemark>` és `</Placemark>` elemek közé beírjuk a

```
<TimeSpan>
  <begin>2007-01-14T21:08:33Z</begin>
  <end>2008-01-14T21:08:33Z</end>
</TimeSpan>
```

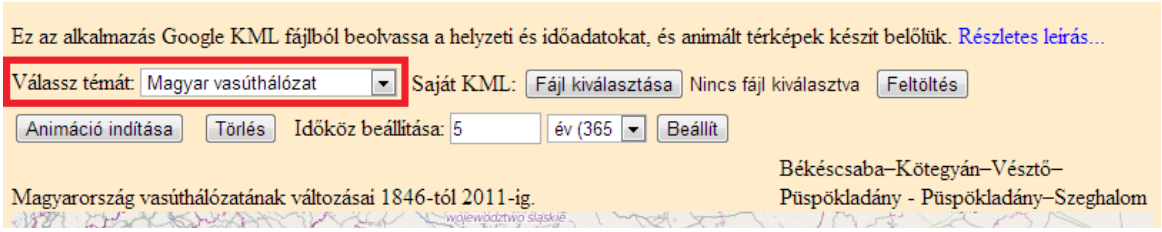
vagy

```
<TimeStamp>
  <when>2007-03-14T21:08:33Z</when>
</TimeStamp>
```

elemeket a megfelelő időpontokkal.

6.3 Adat kiválasztása, feltöltése

A honlapon egy legördülő menüből lehet kiválasztani, hogy melyik állományt szeretnénk megjeleníteni az előre feltöltöttek közül. [6.3. ábra]



6.3. ábra: Egy kiválasztott téma

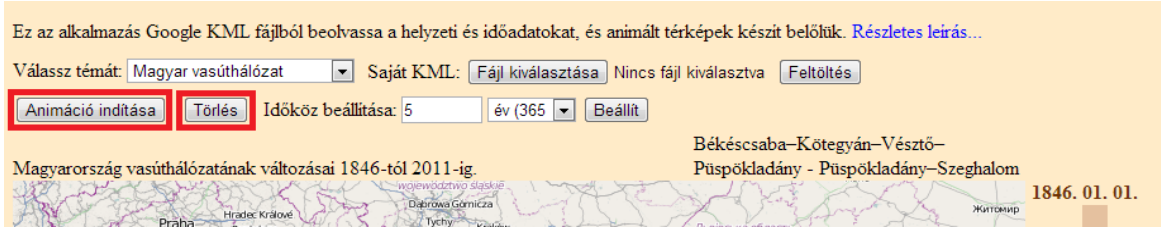
A másik lehetőség, hogy a feltöltés mezővel saját állományt töltünk fel az oldalra. Ez az állomány nem kerül fel a szerverre a többi mellé, csak a feltöltője futtathatja le az animációt, amíg a munkamenet tart, tehát ki nem lép a böngészőből, vagy nem választ másik animációt. [6.4. ábra]

6.4. ábra: Saját feltöltött adat

6.4 Animáció lejátszása, megállítása

A lejátszás az `Animáció indítása` gomb megnyomásával indítható el.

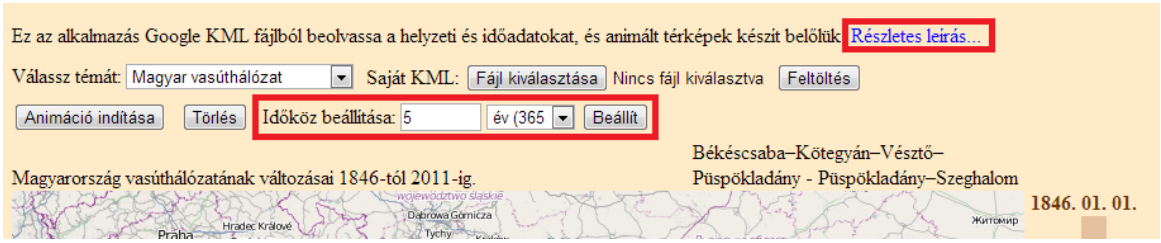
Az animáció leállításhoz a `Törlés` gombot kell megnyomni, vagy ki lehet választani, illetve fel lehet tölteni egy másik állományt. [6.5. ábra]



6.5. ábra: Az animáció vezérlőgombjai

6.5 További funkciók

Az oldal tetején a cím alatt egy rövid ismertetőt olvashat a látogató az alkalmazásról. Ha megnyomja az egy soros leírás mellett a linket, akkor megjelenik egy részletesebb ismertető a funkciókról. [6.6. ábra]



6.6. ábra: Részletes leírás, időköz beállítása

Az `Időköz beállítása` funkcióval meg lehet adni, hogy egy animációs lépésben mennyi idő teljen el. Ehhez minden térkép esetén kiszámít az alkalmazás egy értéket, amit azonban a felhasználó felülírhat. [6.6. ábra]

Egy rajzi elemre kattintva megjelenik a neve a térkép felett jobb oldalon.

7 A rendszer továbbfejlesztésének lehetőségei

7.1 Animáció időzítése, megállítása

Hasznos fejlesztés lenne, ha a felhasználó lehetőséget kapna az animáció megállítására, illetve egy adott időpontra ugrásra. A kezelőfelületen ehhez csak minimális módosítás kellene. Elég volna egy indítás/megállítás gombot felvenni, illetve lehetőséget adni, hogy az idősávra kattintva megtörténjen pozicionálás.

A komolyabb munkát a funkció gép oldali megvalósítása jelentené. Ehhez át kellene kicsit alakítani az adatszerkezetet, hogy könnyebb legyen keresni a már megjelenített, és a még megjelenítendő elemek között. Ezen kívül az időzítés függvényét is át kellene írni, hogy a meghatározott animációs lépésekben állítsa be az időzítőket azokra az elemekre, amiket a következő animációs lépés előtt már meg kell jeleníteni.

7.2 Animáció sebességének módosítása

Jelenleg beállítható ugyan, hogy egy lépésben mennyi idő teljen el, azonban hasznos funkció lenne az is, hogy mennyi időnként történjen léptetés. Ehhez egy gördítősávot lehetne felvenni az oldalra, melynek arrébb tolásával megváltozna a lépések sebessége.

Ennek a funkciónak a megvalósításához is szükséges lenne az előző pontban már felvázolt adatszerkezeti és időzítési átalakítás elvégzése.

7.3 Időadatok kiírása a térképi elemekre kattintva

Megvalósítható lenne, hogy a térképi elemekre kattintva ne csak a nevük, de megjelenítési időadataik is megjelennének. A jelenleg megjelenített név mező magában az SVG-ben kerül tárolásra, azonban további leíró adatok tárolásához már érdemesebb lenne más megoldást találni, például egy JavaScript tömbbe kiírni az adatokat hasonlóan az időzítési adatokhoz.

8 Összegzés, tapasztalatok

Diplomamunkámban röviden áttekintettem a térképi animációk elméletét, majd a weben, böngészőből is elérhető térképi animációs lehetőségeket, előnyeiket és hátrányaikat. Ezen lehetőségek közül a vektorgrafikus, SVG formátumú megjelenítés segítségével, PHP és JavaScript felhasználásával készítettem el saját webes alkalmazásomat.

Még sok funkcióval ki lehetne egészíteni az alkalmazást, azonban úgy gondolom, hogy sikerült bemutatnom egy új, modern lehetőséget a témakörben, melynek előnye, hogy az egyre népszerűbb hordozható eszközökön is jól működik, kényelmesen használható.

Tapasztalatként megemlíteném, hogy bár az alkalmazás valóban működik minden modern böngészőn (Internet Explorer 6, 7, 8 esetén beépülő telepítésével), két hibával azért találkoztam. A kisebbik hiba az Internet Explorer 10-es verziójában jelentkezett. A tényleges tartalmon túl is tart az oldal mérete, aminek következtében megjelenik a vízszintes és függőleges görgetősáv. Ennek az lehet az oka, hogy a böngésző a `viewbox`on kívül eső, tartalommal nem rendelkező SVG koordináta-rendszert is beszámítja az oldal méretébe. Azonban ez a jelenség az esztétikain kívül más problémát nem eredményez. A másik, komolyabb hiba, hogy a Firefox böngésző 17-nél újabb verziói Windows operációs rendszer esetén alapértelmezés szerint egy új 2D-s megjelenítő modult használnak (Azure), ami egy hibából adódóan bizonyos esetekben nem jeleníti meg a pontszerű SVG objektumokat (Bugzilla, 2012). A hibáról, és annak kikerüléséről (Azure kikapcsolása) tájékoztatom az érintett látogatókat.

9 Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Gede Mátyásnak a diploma-munkám elkészítésében fontos szerepet játszó tanácsokért. Köszönöm továbbá José Jesús Reyes Nuñez tanár úrnak a téma szakirodalmának megismerésében nyújtott segítségét, és Szabó Tímeának a weboldal küllemének kialakításában adott tanácsokért.

10 Irodalomjegyzék

Asproth, Viveca (1995): Visualization of Dynamic Information., School of Business, Stockholm University, Stockholm

Bugzilla (2012): Bug 820988 - [Azure] svg object (circles) not displaying after Firefox 17, w/ gfx.content.azure.enabled turned on (which it is by default now) https://bugzilla.mozilla.org/show_bug.cgi?id=820988
Utolsó elérés: 2013. június 1.

European Commission (2013): Portable Network Graphics (PNG). In: Information Provider's Guide – The EU Internet Handbook
http://ec.europa.eu/ipg/standards/image/png/
Utolsó elérés: 2013. június 1.

Harrower, Mark – Fabrikant, Sara (2008): The Role of Map Animation for Geographic Visualization. In: Martin Dodge, Marty McDerby and Martin Turner (Editors): Geographic Visualization – Concepts, Tools and Applications, John Wiley & Sons, West Sussex, pp. 49−65.

Neuendorffer, Steve (2000): CompuServe GIF Image Format.
http://ptolemy.eecs.berkeley.edu/eecs20/sidebars/images/gif.html
Utolsó elérés: 2013. június 1.

OpenStreetMap Wiki (2013): Mercator.
https://wiki.openstreetmap.org/wiki/Mercator
Utolsó elérés: 2013. június 1.

Paul, Ryan (2008): Mozilla drags IE into the future with Canvas element plugin
http://arstechnica.com/information-technology/2008/08/mozilla-drags-ie-into-the-future-with-canvas-element-plugin/
Utolsó elérés: 2013. június 1.

Peterson, Michael P. (1994): Spatial Visualization through Cartographic Animation: Theory and Practice.
http://maps.unomaha.edu/mp/Articles/GISLIS/VisAnim.html.
Utolsó elérés: 2013. június 1.

Peterson, Michael P. (1996): Between Reality and Abstraction: Non-Temporal Applications of Cartographic Animation.
http://maps.unomaha.edu/AnimArt/article.html.
Utolsó elérés: 2013. június 1.

Peterson, Michael P. (2000): Cartographic Animation.
http://maps.unomaha.edu/mp/Articles/CartographicAnimation.html.
Utolsó elérés: 2013. június 1.

PHP (2011): Manual – json_encode.
http://www.php.net/manual/en/function.json-encode.php#105789.
Utolsó elérés: 2013. június 1.

Savarese Software Research Co. (2012): Ssrc SVG: SVG Plugin for Internet Explorer
http://www.savarese.com/software/svgplugin/
Utolsó elérés: 2013. június 1.

Schafer, L. Zale (1993): The Future of Animated Maps.
http://web.archive.org/web/20090516205441/http://maps.unomaha.edu/Peterson/methods/Research/Zale/Future.html.
Utolsó elérés: 2013. június 1.

Tuupola, Mika (2008): Simple Static Maps With PHP.
http://www.appelsiini.net/2008/10/simple-static-maps-with-php.
Utolsó elérés: 2013. június 1.

W3C World Wide Web Consortium (2011): SVG 1.1 (Second Edition) – Coordinate Systems, Transformations and Units.
http://http://www.w3.org/TR/SVG/coords.html.
Utolsó elérés: 2013. június 1.

Winokur, Danny (2011): Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5.
http://blogs.adobe.com/conversations/2011/11/flash-focus.html.
Utolsó elérés: 2013. június 1.

11 Ábrajegyzék

12 Mellékletek

12.1 A mellékelt CD tartalma

diplomamunka_kadari.pdf: A diplomamunka PDF formátumban.

12.1.1 A work könyvtár tartalma

Ezek a fájlok nem érhetőek el közvetlenül az internetről, csak a PHP használhatja őket.

programfájlok:

- **globals.inc.php**: a globális állandókat tartalmazó PHP fájl
- **svg_anim.inc.php**: az alkalmazás működéséhez szükséges PHP függvények gyűjteménye
- **php_google_maps**: külső függvénykönyvtár a hozzá tartozó fájlokkal együtt
- **php_google_maps\staticmap.inc.php**: a háttértérkép sarokkoordinátáit kiszámító függvény

adatfájlok:

- **sourcedata\hazsongardi_temeto.kml**
- **sourcedata\hazsongardi_temeto_old.kml**
- **sourcedata\mav2012.kml**
- **sourcedata\nemzeti_parkok.kml**

12.1.2 A www könyvtár tartalma

Ennek a könyvtárnak a tartalma érhető el az internetről.

- **anim.js.php**: valós időben generálja a JavaScript tömb változót.
- **animsvg.js**: az animációhoz szükséges JavaScript függvények
- **date.js**: dátumkezelő függvénykönyvtár
- **globals_path.inc.php**: a globális állandókat tartalmazó fájlra mutató PHP fájl
- **index.php**: a főoldal
- **page.js**: egyéb, nem animációhoz kötődő JavaScript függvények
- **style.css**: CSS stíluslap az oldal esztétikus megjelenéséhez

Nyilatkozat

Alulírott, …….. nyilatkozom, hogy jelen dolgozatom teljes egészében saját, önálló szellemi termékem. A dolgozatot sem részben, sem egészében semmilyen más felsőfokú oktatási vagy egyéb intézménybe nem nyújtottam be. A diplomamunkámban felhasznált, szerzői joggal védett anyagokra vonatkozó engedély a mellékletben megtalálható.

A témavezető által benyújtásra elfogadott diplomamunka PDF formátumban való elektronikus publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2013. június 7.

…….

(a hallgató aláírása)