Térinformatikai fejlesztések hálózat topológiák alkalmazásával

doktori (PhD) értekezés

Zichar Marianna

Debreceni Egyetem
Debrecen, 2003.

Tartalomjegyzék

„… ha a tudományt műveljük,
nyugodtan állíthatjuk, hogy teszünk
valamit: a talaj itt biztos, s minden
felfedezés, még a legkisebb is,
megmarad.”
Pierre Curie
francia fizikus

1. Bevezetés

A térinformációs rendszerek adatállományát képező bármely objektum rendelkezik egy olyan jellemző tulajdonsággal, mely biztosítja számára a térinformációs rendszerekbe történő integrálhatóságát. Az objektumok (tárgyak, jelenségek) e közös vonása, hogy a helyzetük fontos szerepet tölt be a valós világ modell felépítésének minden fázisában. Annyira domináns e tulajdonság, hogy a térinformációs rendszerek definiálására is felhasználható, miszerint: a hellyel kapcsolatos információk gyűjtésére, feldolgozására és közlésére alkalmas információs rendszereket térinformációs rendszereknek nevezzük [8].

A funkció szerinti négykomponensű rendszer legsokrétűbb, legszínesebb összetevője az adatelemzést megvalósító eljárások alkotják, melyben a rendszer valódi funkcionalitása teljesedik ki. Az adatok elemzését szolgáló funkciók családja a topológiai modellt alkalmazó rendszerekben a legnépesebb, ami egyben azt is indikálja, hogy a legtöbb, legváltozatosabb új információt ezen a modellen alapuló alkalmazások biztosítják. E terület jeles képviselője az Autodesk Map 6-os verziója, melyet AutoCAD környezetben, nagy pontosságú térképezési és térinformatikai elemzések elvégzésére készítettek.

Ezek a tények és az Autodesk Map 6 szoftver megismerése irányította figyelmemet az adatelemzési lehetőségek kutatására. A munkahelyemen két másik csoport által kifejlesztett (nem térinformatikai) alkalmazás hatására végül az adatelemzési funkciók közül is a hálózatelemzési kérdésekkel kezdtem el részletesen foglalkozni. Célom az Autodesk Map 6-os verziójának ide kapcsolódó eljárásainak finomítása, felhasználhatóságának bővítése, illetve a szoftver használata során általam és kollegáim által hiányolt kényelmi szolgáltatások megvalósítása.

A hálózatelemző függvények a topológiát alkotó összes objektumot bevonják a művelet hatáskörébe. Ha a topológia nem homogén összetételű, azaz nem azonos objektum osztályból kerülnek ki az élek, akkor gyakran előfordulhat, hogy nem a teljes topológián kívánnánk a műveletet végrehajtani, ha a lehetőségek engednék. Ennek az igénynek a kielégítésére bevezettem a hálózat résztopológia fogalmát, s gyakorlati megvalósítását is részletesen kidolgoztam.

Munkám eredményeit egy Visual LISP nyelven kifejlesztett térinformatikai alkalmazásba integrálva is bemutatom, mely a betöltött menü menüpontjainak segítségével érhető el Autodesk Map 6 felületen. Az alkalmazás és a szükséges menüállományok (forráskódokkal együtt) teljes terjedelemben megtalálhatóak jelen értekezés CD mellékletén. Az új AutoCAD parancsok VLISP alakját az alkalmazás is számos alkalommal használja, míg az interaktív használatra parancssori változatukat is elkészítettem.

Az alkalmazás újszerű megoldásokat tartalmaz elemek, két listadoboz közötti mozgásának nyilvántartásához, az elemző függvény eredményének riport formájában történő megjelenítéséhez is. A jelenleg kétnyelvű alkalmazás által ismert nyelvek köre utólag is tetszőlegesen módosítható, illetve a grafikus adatbázis is lecserélhető.

Kifejlesztettem és bemutatom még a *pont a poligonban* algoritmus általánosított változatát is, mely a térinformatikában gyakran előforduló bonyolultabb alakzatok esetén is hiba nélkül működik. Az algoritmus VLISP implementációját az alkalmazásba is beépítettem tesztelés céljából.

Minden szakterületnek megvan a saját szaknyelve, mely igen gyakran idegen nyelvterületen (legtöbbször angol nyelvterületen) keletkezik Jó esetben az új kifejezések magyar megfelelőinek megtalálása nem okoz különösebb fejtörést az érintett szakmabelieknek, de sajnos gyakran előfordul, hogy nem lesz általánosan elfogadott magyar megfelelője a kifejezésnek. Ennek okait természetesen nem áll módomban elemezni, következményeit viszont jelen dolgozat is viseli. Ez konkrétan azt jelenti, hogy igyekeztem magyar szavakat használni az értekezésben, ahol csak lehetett, de előfordul, hogy jobbnak láttam az angol megfelelőjének zárójelben való közlését is. Néhány kivételes esetben fordul elő csupán, hogy angol szó lefordítatlan maradt.

„Az ismert tények útvesztőjében
terv nélkül könnyen elveszhet
az ember.”
Mengyelejev
orosz kémikus

2. A kutatási környezet meghatározása

A napjainkban működő térinformációs rendszerek nagy többsége elvi felépítését figyelembe véve két nagy csoport egyikéhez tartozik: vagy a raszter alapú vagy a vektor alapú rendszerekhez. A raszteres rendszer alapeleme a képelem, vagy cella, mely segítségével rekurzív felbontással írhatjuk le az objektumokat. Az értekezés szempontjából a másik típusú rendszerek vizsgálata a hangsúlyos, ezért csak a vektoros rendszerek legfontosabb jellemzőit mutatom be.

2.1. A vektor alapú adatmodellek

Vektor alapú rendszerben a logikai modell alapegységének tekintett objektumok a geometriai adatát vektorok segítségével adjuk meg, s az adatmodell alapegységének pedig a pont és annak koordinátái tekinthetők valamilyen vonatkozási rendszerben meghatározva. A vonatkozási rendszerek biztosítják a geometriai adatok Földhöz történő kapcsolatának leírását. A koordináták tárolásának konkrét megvalósítása, az alkalmazható algoritmusok vektoros modelleken belül is a további altípusok függvényében igen nagy változatosságot mutat.

Vektoros rendszerben két tetszőleges pont összekötésével él definiálható, mely vonalas objektumok reprezentálására használható. Élek kapcsolódó, majd záródó rendszere poligont alkot, ami már területtel rendelkező objektum. Összesen e három elem (pont, él és poligon) felhasználásával írjuk le a valós világot. A vektoros modellen alapuló rendszerek kisebb helyigényűek (a kódtömörebb ábrázolás miatt), a nagyítás, kicsinyítés nem befolyásolja a képminőséget, s grafikailag összességében is jobb képet nyújt a raszteres modellhez képest. Algoritmusai kidolgozottabbak, elterjedtebbek.

2.2. A topológiai adatmodell

A topológiai modell a vektoros adatmodellek családjának egyik legismertebb tagja, s egyben kutatási területem. Elnevezése arra utal, hogy leképezésekkel szemben invariánsan viselkedő szomszédsági információt is tárol. (A geometriai topológia a téralakzatok azon tulajdonságait vizsgálja, melyek nem változnak az idomok szakadásmentes torzítása során.)

A topológiai adatok megadásakor explicit formában külön-külön tárolódik a csomópontok, élek illetve a poligonok topológiája. Ezeket az információkat még az adatbevitel során el kell készíteni, így a modell felépítése igen időigényesnek mondható. Ennek az aprólékos munkának eredményeképpen, viszont nem csak nyilvántartások megvalósítására, hanem testek leírására, térbeli analízisek végrehajtására is alkalmas modellt készítettünk.

Fontos megjegyezni, hogy az objektumok tulajdonságait leíró attribútumokat pontokhoz, élekhez illetve poligonokhoz egyaránt rendelhetünk. Hardver igényét tekintve egy topológiai modellen alapuló alkalmazás erőforrásigénye jelentős mind tárolókapacitás, mind pedig memória felhasználás terén.

A fejlesztésekhez felhasznált, s a következő fejezetben bemutatásra kerülő Autodesk Map is topológiai adatmodellen alapszik.

„A földön 2 milliárd AutoCAD dwg rajzfájl halmozódott fel; az ezredfordulón a világ minden harmadik lakosára jutott egy AutoCAD rajzfájl.”

3. Az Autodesk Map 6 bemutatása

Az Autodesk Map egy AutoCAD alapú specializált térképezési szoftvertermék, amely tartalmazza az AutoCAD program teljes funkcionalitását kiegészítve hatékony, kifejezetten térképészeti és térinformatikai szakemberek részére készített eszközökkel. A felépítését tekintve egy olyan vektoros rendszer, mely topológiai adatmodellen alapszik.

3.1. Alapfogalmak

Az Autodesk Map kétféle dwg állományt különböztet meg [2, 3]. Az AutoCAD környezetből ismert rajzfájlt forrásrajznak nevezi, s bevezeti mellette a projekt fogalmát. A projekt egy olyan rajzfájl, amely a csatolt forrásrajzokhoz, elmentett lekérdezésekhez, alapértelmezett beállításokhoz, külső adatbázisokhoz és csatolási sablonokhoz tartalmaz hivatkozásokat. Ezzel ellentétben a forrásrajz nem tartalmaz ilyen jellegű információkat. A projektben történő felhasználás céljából csatolásra került rajzok összességét rajzkészletnek nevezzük. A projektrajzok egymásba ágyazhatóak, ami azt jelenti, hogy a rajzkészlet részeként – szükség szerint – projektrajzok is megadhatóak. Ebben az esetben a csatolt projektrajz csatolt forrásrajzait beágyazott rajzoknak nevezzük. Egyetlen munkafolyamat során egynél több projekt is megnyitható, de egyszerre csak egy lesz közülük mindig aktív. Egy jól meghatározott feladatnak létrehozható egy projekt, mely egyben leírja a munkakörnyezetünket.

A forrásrajzok csatolásakor meghajtó álnevek (*drive alias*) felhasználásával adható meg az állományok pontos helye. Egyetlen előre definiált alapértelmezett meghajtó álnév van, mely a C meghajtóra mutat. A saját projektek számára külön gondoskodni kell a megfelelő mappák számára álnevek meghatározásáról. Az álnevek használata jelentősen megnöveli a projektek és a projekteket használó alkalmazások

hordozhatóságát, hiszen egy másik számítógépes környezetben csak az álnév megfelelő mappára történő irányításával, azonnal használatba vehető a projekt, illetve futtatható az alkalmazás.

A forrásrajzokban található objektumok (rajz módú) lekérdezések segítségével másolódnak át a projektrajzba, ahol elvégezhetőek a szükséges műveletek. Az esetleges változtatások mentését illetően több megoldás közül választhatunk:

- Mentési halmaz definiálását követően a változtatásokat visszamentjük a forrásrajzokba.
- A projektrajzba mentjük el a műveletek eredményeként előállt eltéréseket. (Ebben az esetben az eredeti objektumok változás nélkül rendelkezésünkre állnak a forrásrajzokban.)
- Új projektrajzba mentünk.
- A teljesség kedvéért megemlíthető, hogy a változtatások el is vethetőek.

Egy objektum jellemzőit a Tulajdonságok ablakban tekinthetjük meg táblázatos formában, felhasználóbarát környezetben. A valóságban az objektumot leíró adat egy asszociációs listában tárolódik a rajzfájlban, melynek legkisebb elemei olyan speciális kételemű listák, ahol az első elem az adatot azonosító kód, míg a második elem az adattípus, objektumra vonatkozó konkrét értéke.

((-1 . <Entity name: 40096a30>) (0 . "LWPOLYLINE")
(330 . <Entity name: 40083cf8>) (5 . "328E") (100 . "AcDbEntity") (67 . 0)
(410 . "Model") (8 . "Utca_ny") (48 . 0.05) (370 . 20) (100 . "AcDbPolyline") (90 . 5)
(70 . 0) (43 . 0.0) (38 . 0.0) (39 . 1.0) (10 21.5476 37.0103) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 22.0805 34.8448) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 23.9633 37.5783) (40 . 0.0)
(41 . 0.0) (42 . 0.0) (10 24.8869 36.9393) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 24.212 35.8743) (40 . 0.0) (41 . 0.0) (42 . 0.0) (210 0.0 0.0 1.0))

A –1-es kóddal az objektum azonosítója figyelhető meg, mely egy olyan numerikus címke, ami az állomány megnyitásakor keletkezik újonnan. Ennél a tulajdonságánál fogva nem alkalmas program szintű tárolásra, nem úgy, mint például a fogó értéke, mely állandó jellemzője egy objektumnak (5-ös kód), annak élete során nem változik. A teljesség kedvéért viszont meg kell jegyeznünk, hogy használata letiltható. Az objektum típusát 0-s kód jelzi, míg a tartalmazó fóliát például a 8-as. A fentebb bemutatott

asszociációs lista egy vonalláncé, melyet meghatározó pontok koordinátái 10-es kóddal kerülnek felsorolásra. (Jelen esetben öt pont definiálja a vonalláncot, amiről egyébként a 90-es csoportkóddal jelölt lista is árulkodik.)

Az alkalmazások által, illetve a dxf fájlformátumban használt csoportkódok részletezése jelentősen meghaladja e dolgozat terjedelmét, ezért csak ízelítőnek tekinthető az előző bekezdés. Ahol feltétlenül szükséges, ott mindig közlöm a probléma, vagy a körülmények megértéséhez elengedhetetlen információkat, azzal a megjegyzéssel, hogy az asszociációs listák teljes körű leírása a Developer Documentation súgó fájl DXF Reference fejezetében található meg [1].

3.2. Attribútum adatok tárolásának lehetőségei

A térinformációs rendszer felépítésében résztvevő objektumok a geometriai adatokon túl rendelkezhetnek úgynevezett szakadattal, vagy más néven attribútum adattal is. Az objektum, alkalmazás szempontjából fontos jellemzőit tároljuk ilyenformán, amelyek a különböző műveletek (lekérdezések, tematikus térképek készítése) során kerülnek majd felhasználásra. Ebben a fejezetben az Autodesk Map környezet által az attribútum adatok definiálására, tárolására biztosított két, lényeges különbséggel rendelkező lehetőség kerül áttekintésre.

3.2.1. Objektumadat táblák

Objektumadatok használatához Autodesk Map felületen előbb definiálnunk kell a tábla szerkezetét, azaz meg kell határoznunk a táblát alkotó mezők főbb jellemzőit (név, leírás, típus, alapértelmezett érték). Egy új rekord mezőinek kitöltését követően, a rekord objektumhoz, vagy objektumokhoz csatolható. Egy táblából több rekordot is csatolhatunk ugyanahhoz az objektumhoz, így akár az 1:N típusú kapcsolat is megvalósítható. (A kapcsolat típusa a párbeszédpanelben egy jelölőnégyzettel szabályozható.) Egy objektumadat táblában rögzített rekordokat nem lehet egyszerre, például táblázatos formában áttekinteni, csak szigorúan az objektumok kijelölésén keresztül. (Egy objektum kijelölését követően viszont annak bármely táblában definiált rekordja megtekinthető vagy szerkeszthető a helyzettől függően.)

((-1 . <Entity name: 40098b88>) (0 . "LWPOLYLINE") (5 . "EA1")
(102 . "{ACAD_XDICTIONARY") (360 . <Entity name: 40067a90>) (102 . "}")
(330 .<Entity name: 4006ccf8>) (100 . "AcDbEntity") (67 . 0) (410 . "Model")
(8 ."Utca_halozat") (370 . 20) (100 . "AcDbPolyline") (90 . 2) (70 . 0) (43 . 0.0)
(38 . 0.0) (39 . 1.0) (10 23.5588 49.4716) (40 . 0.0) (41 . 0.0) (42 . 0.0)
(10 25.8594 50.4009) (40 . 0.0) (41 . 0.0) (42 . 0.0) (210 0.0 0.0 1.0))

Az objektumadat tábla magában a rajzfájlban kerül tárolásra, növelve ezzel annak méretét. Egyrészt az objektumokat leíró asszociációs lista is hosszabb lesz (a második sorban a 102-es kóddal kezdődik és fejeződik be az objektumadatra utaló rész), másrészt nem grafikus objektumként szintén a rajzfájlban kerülnek rögzítésre a tábla adatai. Amennyiben fontos tényező a rajzállomány mérete, nagy mennyiségű adatot nem ajánlott objektumadatként tárolni. Rajz objektumot még nem tartalmazó rajzban definiálható ugyan objektumadat tábla, de rekord csak akkor rögzíthető, ha azonnal objektumhoz is csatoljuk. Ennek egyenes következménye, hogy egy objektum törlésekor a hozzá csatolt adat is elvész (két módon is csökkentve ezzel fájl méretét), illetve az hogy, nincs lehetőségünk automatizálni az adatokat tartalmazó rekordok objektumokhoz való csatolását.

Külső alkalmazások számára az objektumadatok nem hozzáférhetőek, megadásukra, lekérdezésükre, módosításukra, törlésükre csak Autodesk Map felületen nyílik lehetőség. Előnyként kell viszont megemlíteni, hogy az objektumadatokhoz való hozzáférés egyszerű, külső tényezőktől kevésbé függ, s a korábban definiált táblák szerkezetének módosítása, a táblák átnevezése is igen rugalmasan kezelhető. Ritkán változó, nem nagy mennyiségű információ tárolására ideális megoldást jelent.

3.2.2. Külső adatbázisok használata

Az attribútumok definiálásának másik lehetséges módja a külső adatbázisokból származó információk felhasználásában rejlik. Az Autodesk Map kapcsolódni tud Microsoft Access-ben, dBase-ben, Oracle-ben, FoxPro-ban, Paradoxban, vagy bármilyen más ODBC-rendszerű adatbáziskezelő rendszerben készült adatbázisokhoz [2]. Eredeti formájában használja az adatforrásokat, sőt az esetleges változásokat is eredeti formájába menti vissza, így gyakorlatilag a rajzfájl méretét számottevően nem növeli. Ugyanaz az adatbázis tábla több rajz, vagy akár más

alkalmazások által is használható, így például az adatok frissítését végző személyeknek nem feltétlen szükséges Autodesk Map környezetet biztosítanunk.

A külső adatbázisokban tárolt adatok használatba vétele kétlépcsős folyamat. A csatolás során az adatbázishoz már meglévő, vagy frissen létrehozott udl állomány segítségével kiépül a kommunikációs csatorna, mely a kapcsolódás során nyílik meg. Fontos megjegyezni, hogy a kapcsolódás megszüntetése a csatolás megszűnését nem vonja maga után. Alapértelmezett esetben az udl állományok a Data Links mappában találhatóak, nem pedig a projektet tartalmazó mappában. Így amennyiben a projekt külső adatbázisokat is használ, úgy a projekt mozgatásakor külön figyelmet kell fordítani az udl fájlok megfelelő helyen való jelenlétének biztosítására.

Az Adatnézet ablak segítségével (még akár rajzobjektumokhoz való csatolás előtt) lehetőség nyílik a rekordok szimultán áttekintésére, szűrésére, rendezésére, nyomtatására vagy vágólapra helyezésére. Sok esetben új rekorddal is bővíthető az adatbázis, s rekordok törlése is elvégezhető. (A szerkeszthetőség a megnyitás módján kívül az adatforrás típusától, illetve írásvédettségétől függ.) Az adatbázis egy kulcsnak alkalmas mezőjének, vagy mezőcsoportjának kiválasztásával csatolási sablon határozható meg, mely a grafikus objektum és az adatbázis tábla közötti kapcsolatot írja le. Egy csatolási sablon definiálását követően manuálisan, de akár automatikusan is elvégezhetjük a rekordok rajzobjektumokhoz való rendelését.

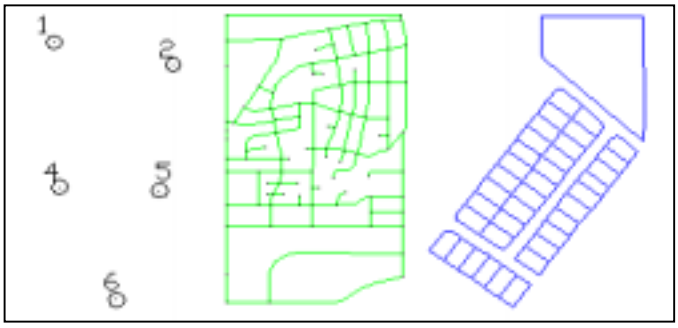
```
(-3 ("DCO15" (1071 . 4) (1071 . 4) (1000 . "utca_css") (1004 . "074D343200")))
```

Az objektumot leíró asszociációs lista csak egy úgynevezett csatolási adattal bővül, ami alapvetően a csatolási sablon nevéből (1000-es kód) és a kulcsnak definiált mező(k) kódolt aktuális értékéből áll (1004-es kód). Így egy objektum törlésekor csak annak csatolási adata vész el, a külső adatbázis rekordjai érintetlenek maradnak. Egy objektumhoz több rekord is csatolható, s ez fordítva is így van, azaz egy rekord több objektumhoz is hozzárendelhető, azaz N:M típusú kapcsolat is megvalósítható. A Tulajdonságok ablak megjeleníti a külső adatbázisból az objektumhoz csatolt rekordokat, ami igen kényelmes szolgáltatásnak mondható.

Mint láttuk egy külső adatforrás használatához több tényezőre kell párhuzamosan figyelni, de ez a többlet energia megtérül az előnyös tulajdonságai révén.

3.3. Topológiák

A topológiai modellben értelmezhető összes topológia felépíthető Autodesk Map-ben is. A három térkép-alapú topológia (csomópont, hálózat és poligon topológia) háromfajta grafikus elemből épül fel: csomópontok, élek és poligonok halmazából. A topológiák magán a térképen hozhatók létre, ott lehet módosítani és lekérdezni is őket. A topológia felépítésében résztvevő minden elem objektumadatként tárolja a topológiai információit.



1. Ábra Csomópont, hálózat és poligon topológia

A topológiai adatok létrehozásának igen fáradságos részét a program precízen elvégzi helyettünk, amennyiben minden paramétert helyesen megadtunk a megfelelő párbeszédpanelben. Így például minden csomópont, él és poligon automatikusan kap egy egyedi azonosítót, amire a különböző műveletek végrehajtása során szüksége van a rendszernek. A topológiai adatokat tartalmazó objektumadat táblák mezőinek nagy része nem szerkeszthető, s a táblák sem bővíthetőek további mezőkkel (noha gyakran jó szolgálatot tenne). Számos topológián illetve topológiák között értelmezett műveleti lehetőség áll rendelkezésünkre, melyek eredménye is igen sokfélék lehetnek. További részletek a Súgóban megtalálhatóak, én a következő részekben a hálózat topológiát és annak műveleteit részletezem, melyre vonatkozóak a saját eredményeim is.

3.4. A hálózat topológia

A hálózat topológia a lineáris hálózatot alkotó élek kölcsönös összefüggéseit írja le (lásd [2]). Két logikai alapeleme a csomópont és az él, melyek felhasználásával fa struktúrájú hálózat éppúgy felépíthető, mint hurkokat is tartalmazó. Értelmezhető a csomópont topológia kibővítéseként is, miszerint a már meglévő csomópontokat kiegészítve őket összekötő élekkel hálózat topológiához jutunk. Hálózat topológiával jeleníthetők meg többek között egy település utcái, a közutak, különféle csővezetékek és akár a folyók is.

3.4.1. Rajz letisztázása hálózat topológia készítéséhez

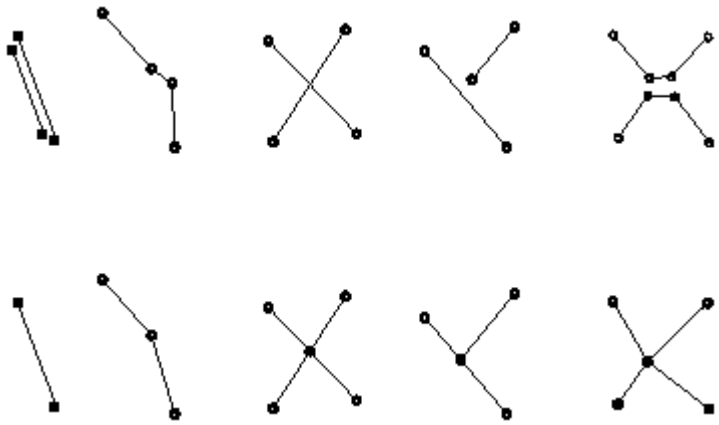
A topológia létrehozását – csomópont topológia kivételével –a rajz letisztázásának mindig ajánlott megelőznie. A térképek letisztázása során a digitalizálásból, szkennelésből, pontatlan rajzműveletek végzéséből eredő hibák szüntethetők meg, lehetővé téve ezzel a kívánt topológia hiba nélküli felépítését. Ezek az eszközök használhatók még a felesleges térképi részletek eltávolítására és a szomszédos térképszelvények pontos illesztésére is (lásd [2]).

Első lépésként a művelet hatáskörébe eső objektumokat kell kiválasztani. Az első párbeszédpanelben állítható be, hogy mely réteg(ek)ről válassza ki az összes vonalas objektumot a program, vagy az objektumok kézi kiválasztása mellett döntöttünk. A második lépés során a tisztázási műveletek listájából a végrehajtandókat kell kijelölni, illetve azok estleges paramétereit meghatározni. Végül a változtatások végrehajtásának alábbi módjai közül választhatunk:

- Eredeti objektum módosítása
- Eredeti objektum meghagyása mellett új objektum készítése
- Eredeti objektum törlése és új objektum készítése

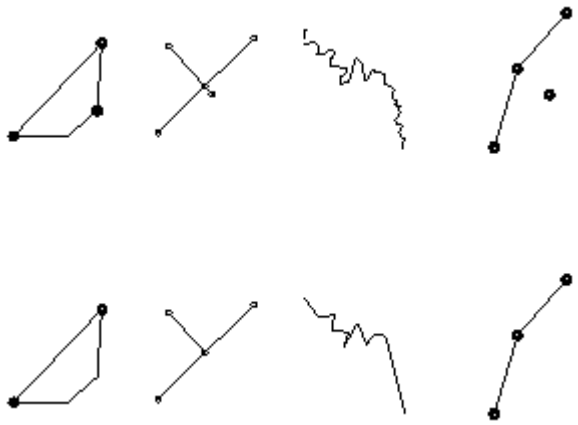
A következőkben az Autodesk Map által értelmezett tisztázási műveletek, és rövid meghatározásukat ismertetem. (Bővebb leírásért a Súgóhoz lehet fordulni.)

A 2. és a 3. ábra felső sora az eredeti objektumokat, az alsó pedig a tisztázási művelet végeredményét mutatja, a műveletek leírásának sorrendjében.



2. Ábra Az oldalon ismertetett tisztázási műveletek szemléltetése

- Kettőzött objektumok törlése.
  Ugyanazzal a kezdő-, és végponttal rendelkező objektumok egyikének törlését jelenti. Ha a két objektum vonalainak különböző az irányuk, attól még kettőzöttnek számítanak, és egyikük törlésre kerül.
- Rövid objektumok törlése.
  Az adott tűrésnél rövidebb objektumok törlése.
- Metsző objektumok megtörése.
  Az egymást nem csomópontban metsző objektumokat megtöri, és elhelyezi a hiányzó csomópontot.
- Hézagok megszüntetése.
  Ha a megadott tűrési sugáron belül egymást megközelítő, de nem érintkező objektumok közül az egyik kiterjeszthető úgy, hogy az messe a másikat, akkor az hozzákapcsolódik az adott tűrésen belül lévő csomóponttal. Ha nincs csomópont, akkor az Autodesk Map létrehoz egyet.
- Csomópontfürtök összevonása.
  Az egymástól a tűréshatárnál kisebb távolságra lévő pontokat, a közülük leginkább középre eső csomóponthoz csatolja.



3. Ábra Az oldalon ismertetett tisztázási műveletek szemléltetése

- Álcsomópontok feloldása
  A csak két objektumhoz tartozó csomópontokat megszüntetve egyesíti a két objektumot.
- Szabad objektumok törlése
  A legalább egyik végponttal objektumhoz nem csatlakozó objektumok törlése.
- Objektumok egyszerűsítése
  Az összetett vonalas objektumok saroktűrésen belül eső csomópontjainak törlés.
- Nullahosszúságú objektumok törlése
  A kezdő-, és végponttal rendelkező, de 0 hosszúságú, illetve a csak kezdőponttal rendelkező vonalláncok kerülnek törlésre a rajzból.

Lehetőség van a kiválasztott műveletek interaktív végrehajtására is, mely során minden egyes változtatás előtt módunkban áll eldönteni, hogy végre kívánjuk-e hajtani az adott objektumon a felajánlott változtatásokat vagy sem.

3.4.2. Hálózat topológia definiálása

A hálózat topológia két logikai alapelemből építkezik: élekből és csomópontokból. Létrehozása során csomópontok újonnan is létrejöhetnek, azonban az élként felhasználni kívánt grafikus objektumok jelenlétéről még a topológia definiálása gondoskodnunk kell.

Csomópontnak választható az AutoCAD pont objektumán kívül a felhasználó által definiált tetszőleges felépítésű blokk is, illetve csomópontok generálhatóak létrehozás közben az él, csomóponttal még nem rendelkező végpontjainál is. Újonnan létrehozandó csomópontok célfóliáját (akár újat is) és típusát (ACAD pont, fehasználó által definiált blokk) előre meg kell adni a megfelelő párbeszédpanelben.

Élek keletkezhetnek már létező vonalakból, vonalláncokból és ívekből egyaránt. (Az említett elnevezések mindegyike AutoCAD fogalom.) Elhelyezkedésüket tekintve csomópontot és élet egyaránt választhatunk több fóliáról, akár manuálisan is. Ugyanakkor meg kell jegyeznünk, hogy lezárt, lefagyasztott illetve kikapcsolt fólia objektumai nem elérhetőek topológia készítés céljából, viszont a le nem fagyasztott papírtérbeli objektumok bekerülnek a topológiába.

3.4.3. Topológia adatai

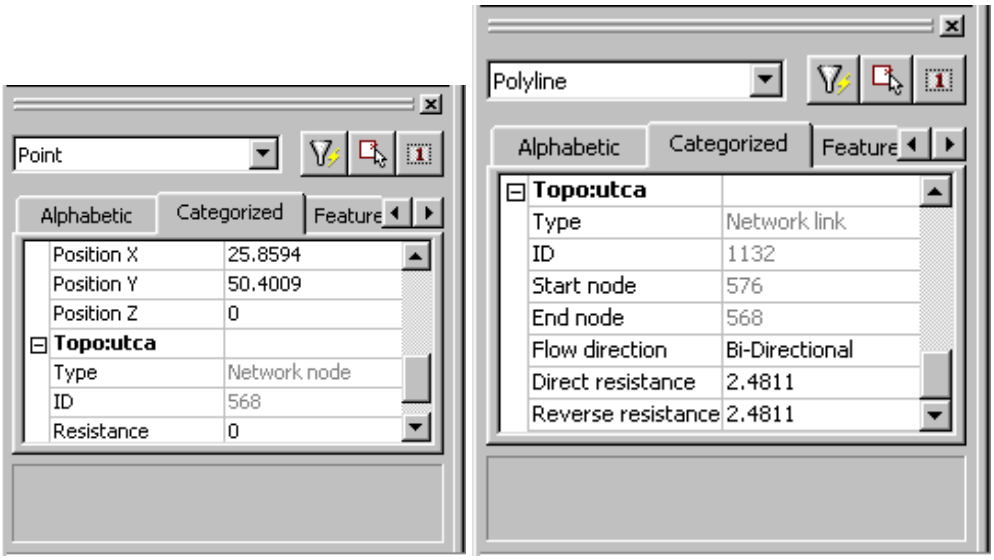
Az elkészült topológia azonnal megjelenik a Munkatér Intézőben, státuszát tekintve memóriába betöltött állapotban. Az újonnan létrehozott objektumadat táblák tartalmát könnyen megfigyelhetjük, akár a Tulajdonságok ablak segítségével, akár az Edit Object Data párbeszédpanel segítségével. A 2. ábrán jól látható, hogy a szigorú értelemben vett topológiai adaton felül, milyen adattal bővül minden csomópont, illetve él objektum.

Csomópont esetén egy ilyen mező van (*Resistance*), melyben tárolt numerikus érték segítségével a csomóponton való áthaladást súlyozhatjuk. Éleknél irány (*Flow direction*), direkt ellenállás (*Direct Resistance*) és fordított ellenállás (*Reverse Resistance*) mezők is megtalálhatóak a topológiai adatok között, melyekkel rendre az áthaladás irányát, illetve az irány függvényében az áthaladás költségét tárolhatjuk. A topológia felépítése során a rendszer alapértelmezett értékekkel tölti fel az előbb említett mezőket, melyeket az 1. táblázat ismertet.

	Alapértelmezett érték
Csomópont ellenállása	0
Él iránya	Kezdőponttól végpontig.
Él ellenállása	Az él hossza.
Él direkt ellenállása	Az él hossza.

1. Táblázat Hálózat topológia szerkeszthető adatainak alapértelmezett értékei

Egy él előre mutató iránya, már az objektum megrajzolásával kijelölésre kerül, s nem a topológia készítésekor keletkező Kezdő pont (*Start node*) és Végpont (*End node*) által van meghatározva. Ezt igen fontos hangsúlyozni, mert általában a topológia generálása előtt jóval korábban keletkeznek azok az objektumok melyekből majd az élek keletkeznek, s ekkor kell lehetőség szerint azonnal a helyes irányítottságot megalapozni. (Természetesen, ha minden él kétirányú lesz, az előbb említettekre nem szükséges odafigyelni, bár általános jó tanácsként érdemes megfontolni, hogy az éleknek szánt objektumokat mindig tudatosan készítsük.)



4. Ábra Csomópont és él objektumadatai hálózat topológiában

A 4. ábrán az is jól megfigyelhető, hogy akár a Tulajdonságok ablakban megváltoztathatjuk a táblázatban felsorolt objektumadatok alapértelmezett értékeit. Ez sokszor kényelmesebb és gyorsabb, mint párbeszédpanelen keresztül. Egy él topológiai adata kiterjesztett adatként (*extended data*) tárolódik, elhelyezkedését tekintve az általános definíciós adatok után, "IRD" alkalmazás kóddal megjelölve következik. (A 4. Ábrán látható él objektum összes topológiai adata az 1004-es csoportkódot követő hexadecimálisan kódolt értékként tárolódik.)

(-3 ("IRD" (1002 . "{") (1000 . "TPMLINK_utca-_-ade") (1070 . 515) (1071 . 1) (1004 . "01006C040000400200003802000000000000033033ED748D9034033033ED748D90340") (1002 . "}")))

Ha a topológia felépítésében résztvevő objektumok forrásrajzból származtak, akkor el kell döntenünk, hogy a projektben létrehozott topológiát vissza kívánjuk-e menteni a forrásrajzba. Nemleges válasz esetén a topológiát hordozó objektumokat el kell menteni a projektrajzba. Érdekes szituáció az, amikor mind a forrásrajz mind a projektrajz tartalmazza a topológiát. (Automatikus mentés mindig a projektrajz állapotát menti, így ha nem figyelünk tudtunk nélkül is könnyedén szembetalálhatjuk magunkat egy ilyen helyzettel. Természetesen ez elkerülhető, hiszen az automatikus mentés letiltható a Tools menü Options párbeszédpanel Open and Save lapján.) Ilyenkor a betöltetlen topológia betöltésekor rendelkeznünk kell, hogy honnan kívánjuk az adatokat felhasználni.

3.4.4. Topológia szerkesztése, törlése

Topológia definiálása után, a topológiát felépítő grafikai objektumokat nem szabad a hagyományos műveletekkel szerkeszteni, ha mégis akkor megsérül a topológia integritása, s könnyen véglegesen használhatatlanná válik. Az Autodesk Map külön topológia szerkesztő parancsokat tartalmaz, melyek használatával biztonságosan elvégezhetőek a kívánt módosítások. Ezen parancsok egyik része a Munkatér Intéző helyzet-érzékeny helyi menüjéből érhető el, míg másik része – valószínűleg biztonsági okokból – csak parancssoron keresztül. A legfontosabb topológia szerkesztő parancsok:

- Új él beszúrása
- Új él hozzáadása

- Él mozgatása
- Él végpontjának mozgatása
- Élek egyesítése
- Él megtörése
- Élek törlése
- Él irányának megfordítása

A topológiák átnevezhetőek, egyszerre csak egy lehet szerkeszthető belőlük a projektben, s a topológia több felhasználó által történő egyidejű szerkesztését sem engedélyezi a program (lásd Súgó és [2][3]). Topológia törlésekor kezdeményezhetjük a geometriai objektumok törlését is, illetve ha a topológia a projektben is és egy forrásrajzban is megtalálható, akkor a projektbeli törlés a forrásrajz topológiáját is törli, hacsak nem inaktív az érintett forrásrajz.

3.5. Hálózat topológiát elemző műveletek

A funkció szerinti négykomponensű térinformációs rendszerek legnagyobb érdeklődésre számot tevő részének az adatelemzés tekinthető. Az adatelemzés lényegében mindig két alapvető lépésből tevődik össze (lásd [8]).

1. Az adat kiválasztása, mely az objektumok geometriai helyzete, vagy attribútumai alapján is történhet.
2. A megfelelő elemzési művelet elvégzése.

Az Autodesk Map háromféle hálózatelemző műveletet definiál, melyek végrehajtásához szükséges paraméterek meghatározása hasonló módon történik. Az elemző műveletek végrehajtása előtt mindig be kell tölteni az elemezni kívánt topológiát, mert csak a betöltötteket ajánlja fel a Topology Selection párbeszédpanelben. Ezt követően négy párbeszédpanel segítségével határozható meg pontosan az elemző művelet, melyeket röviden az alábbiakban ismertetek. (A párbeszédpanelek kitöltésének sorrendje tetszőleges.)

- Elemző művelet típusának kiválasztása
  A következő három alfejezetben bemutatott művelet kiválasztásával a további párbeszédpanelek felépítése is változik.

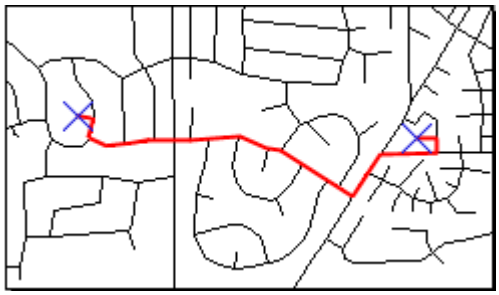
- Csomópontok megadása
  A kiválasztott művelt típusától függően egy vagy több csomópontot kell kiválasztani a rajzterületről. (A kiválasztást lehet módosítani is.)
- Ellenállás és irány paraméterek beállítása
  Amennyiben nem a 3.4.3. fejezetben ismertetett alapértelmezett topológiai adatokat kívánjuk használni az él direkt, fordított ellenállásának, és irányának, illetve a csomópont ellenállásának, akkor tulajdonságokon, topológiai adatokon, külső adatbázison vagy objektumadatokon alapuló tetszőleges kifejezéseket készíthetünk az említett négy mező számára. Itt adható meg a művelet minimális és maximális ellenállása is.
- Eredmény típusának meghatározása
  A művelet eredményét megjeleníthetjük tetszőleges színnel, mint ADETOPVIEW objektum, vagy készíttethetünk új topológiát is belőle. Ez utóbbi esetben természetesen nevet és leírást is kell választanunk.

E számos beállítást természetesen lehetőségünk van elmenteni is, amely során egy úgynevezett profile fájl keletkezik tpf kiterjesztéssel. Az ilyenformán, külső állományban rögzített beállításokat később vissza lehet tölteni, és újra végre lehet hajtani a műveletet. Ha a művelet eredménye topológia formájában elmentésre kerül, akkor két új objektumadat táblával bővül a projekt. Az egyik az újonnan keletkezett topológia hagyományos adatait tartalmazza, míg a másik az útvonal jellemzőiről tárol adatot az élek segítségével.
Az élekhez csatolt rekordok mezőinek tartalma:

- Él sorszáma az út során.
  Ha egy él, többször is felhasználásra kerül az út során, akkor értelemszerűen több csatolt rekordja lesz.
- A direkt ellenállást leíró kifejezés adott élre vonatkozó értéke.
- A fordított ellenállást leíró kifejezés adott élre vonatkozó értéke.
- Kezdőpont kiértékelt ellenállása.
- Végpont kiértékelt ellenállása.
- Az él iránya.

3.5.1. Legrövidebb út

Két csomópont között, a hálózat topológia élein haladva a legrövidebb utat lehet meghatározni, vagy az irány és ellenállás értékeinek megfelelő beállításával az optimális útvonalat. A művelet csak két pont között értelmezhető, és olyan utat határoz meg, mely összellenállása meghaladja a megadott minimum ellenállás értékét, de a maximum ellenállás értéke alatt marad. (Ez utóbbi megállapítás a másik két műveletre is helytálló.)

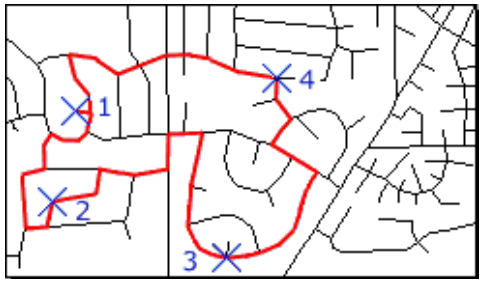


5. Ábra Két pont között értelmezett legrövidebb út

Ez a művelet adhat választ olyan kérdésekre, mint hogyan juthatunk el az otthonunktól az iskoláig, vagy a kórháztól a baleset helyszínéig.

3.5.2. Legjobb út

Egy kiindulási pontnak választott csomópontból, további előre kiválasztott pontok érintésével, a legrövidebb olyan út meghatározása mely visszatér a kiindulási pontba. (A közbenső csomópontok ellenállása, és az élek direkt és fordított ellenállását, irányát természetesen figyelembe veszi.)

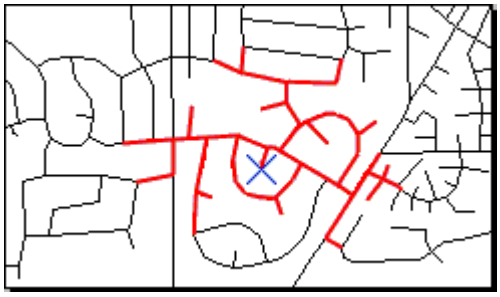


6. Ábra Legjobb útvonal 3 közbenső ponttal

Megterveztethetjük a szállodából induló, és oda visszatérő városnéző sétánkat a meglátogatandó műemlékek kiválasztása után.

3.5.3. Áramlás-elemzés

Az áramlás-elemzés egy kiválasztott csomópontból kiinduló összes olyan utat határozza meg, mely a megadott maximális ellenállás értékét nem haladja meg. Az áramlás-elemző vizsgálat egy hálózat topológia integritásának ellenőrzésére is felhasználható. (lásd [2])



7. Ábra Áramláselemzés végeredménye

A szállodától maximum 500 méteres sétára található látványosságok, s a hozzájuk vezető út kereshető meg ezzel a művelettel.

Az említett elemzések közös, de nem előnyös tulajdonsága, hogy az elemzés hatáskörébe bevont objektumok halmaza nem változtatható dinamikusan. Hasznos lenne az is, ha az ellenállás értékeket meghatározó kifejezések egyszerű elágazásokat is tartalmazhatnának.

3.6. További felhasználási lehetőségek

A topológiák számos művelet végrehajtásában játszhatnak szerepet, melyek teljes körű ismertetése meghaladja jelen dolgozatban a témának szentelhető részt. A kutatási területemhez legközelebb esőket azonban röviden áttekintem ebben az alfejezetben.

Puffervizsgálat végrehajtásával a topológiát alkotó elemektől egy meghatározott távolságon belül elhelyezkedő objektumok találhatóak meg. Ezt a távolságot eltolásnak, vagy a puffervizsgálat sugarának nevezzük. A kiindulási topológia típusától függetlenül a vizsgálat eredménye mindig egy

poligon topológia lesz. A puffereltolás értékének nem csak konstans érték adható meg, hanem tulajdonságokon, objektumadatokon, külső adatbázis rekordokban tárolt adatokon alapuló kifejezés is. (Poligon topológia esetén az eltolás értéke negatív is lehet, ekkor a poligon területén kerül kijelölésre a puffer.)

Topológia lekérdezésével a betöltött topológia és csatolt adatai kereshetőek vissza a projektből vagy egy csatolt aktív rajzból. A hagyományos lekérdezési feltételeknek mind a négy típusa használható, négy lényeges különbség azonban van a hagyományos objektumok és a topológiát alkotó objektumok lekérdezése között:

- A topológiai lekérdezések csak egy topológiára vonatkoznak, míg a hagyományos lekérdezések az aktív csatolt rajzok összes objektumát vizsgálják.
- A topológiai lekérdezéseknél a topológiai adatokat tartalmazó objektumadat táblák bizonyos mezőit is fel lehet használni. (Hálózat topológia esetén például az ellenállás értékeket, és az irányt.)
- A tulajdonság-változtatások poligon topológiákkal másképpen működnek. (lásd Súgó)
- Topológia riport módú lekérdezése esetén két pontváltozóval bővül a riport sablonjába választható pontváltozók köre (.TOPONAME ,.TOPOTYPE). Három pontváltozó pedig eltérő értékkel rendelkezik topológiai lekérdezésekben (.DRAWING, .AREA, .PERIMETER).

Topológiai lekérdezés eredménye tárolható ideiglenes, vagy állandó topológiában is. A lekérdezést definiáló beállítások a hagyományos lekérdezés beállításaihoz hasonlóan a rajzba, vagy külső állományba elmenthetőek.

3.6. Az Autodesk Map testreszabási lehetőségei

Az Autodesk Map az AutoCAD térinformatikai célú kiterjesztéseként jött létre, azaz tartalmazza az AutoCAD teljes funkcionalitását. Ebből kifolyólag a következő felsorolásban megtalálható, az AutoCAD számára kifejlesztett programozási felületek, testreszabási

módszerek mindegyike felhasználható ugyanilyen célból Autodesk Map környezetben is.

- Parancsok testre szabása
- Vonaltípusok, vonalmintázatok definiálása
- Alakok és betűtípusok definiálása
- Egyéni menük készítése
- Script–ek és diák használata
- Programozási felületek
  - Visual LISP/ADSRX
  - ActiveX Automation/ VBA Interface
  - ObjectARX
- Programozható párbeszédpanelek
- PostScript támogatás

A továbbiakban csak a dolgozat későbbi részében általam felhasznált lehetőségeket részletezem kicsit bővebben.

3.6.1. Egyéni menük készítése

A már létező menük is átszerkeszthetőek, vagy teljesen újak is létrehozhatóak tökéletesen igazodva a felhasználói igényekhez. A menük külső megjelenését, és a kiválasztott menüpontokhoz rendelt makrókat úgynevezett menü állományok írják le. A menü makrók lehetnek:

- AutoCAD parancsok
- Feladathoz rendelt billentyű kódok.
- AutoLISP parancsok.
- DIESEL program kódok.
- Vagy ezek kombinációi.

A menü állományok kifejezés mögött valójában hat különböző kiterjesztéssel rendelkező fájl együttesét értjük, melyekről a 2. táblázat ad rövid áttekintést [1].

A hat állomány közül az mns és az mnl kiterjesztésűek szerkeszthetőek közvetlenül VLISP környezetben, vagy tetszőleges szövegszerkesztő segítségével. Az elkészült menü első betöltése az mns állomány segítségével történik, majd az ezt követő első AutoCAD Map

indításkor létrejönnek a szükséges további állományok. A menüket tovább lehet csoportosítani aszerint, hogy a képernyő mely területén, milyen esemény hatására jelennek meg. Ez alapján megkülönböztetünk [1]:

- Mutató eszköz gomb menü.
- Legördülő menü.
- Eszköztár.
- Image tile menü
- Képernyő menü
- Digitalizáló tábla menü
- Súgó üzenetek és eszköztippek
- Billentyűzet gyorsítók

Az eredményeimet bemutató program funkcióinak eléréséhez saját legördülő menüt, eszköztárat, és súgó üzeneteket készítettem, melyek forráskódjai a CD mellékleten megtekinthetőek. A nem részletezett témákban bővebb információval az AutoCAD Developer Documentation [1] szolgál.

Fájl típusa	Leírása
MNU	Sablon menü állomány.
MNC	Lefordított menü állomány, mely bináris formában tárolja a parancsok karakterláncait, és a menü megjelenését.
MNR	Menü forrás állomány, mely bináris formában tárolja a menü által használt bitképeket.
MNS	Forrás menü állomány (AutoCAD által generált).
MNT	Menü forrás állomány, mely nem elérhető mnr állomány esetén keletkezik.
MNL	Menü LISP állomány. A menü állomány által használt AutoLISP kifejezések tartalmazza, melyek memóriába töltődnek az ugyanilyen nevű menü állományok betöltésével egyidejűleg.

2. Táblázat Menü állományok leírása

3.6.2. A Visual LISP fejlesztői környezet és programnyelv

Hosszú éveken keresztül az AutoLISP nyelv szolgált az AutoCAD funkcionalitásának növelésére. Az alapokat képző LISP programozási nyelvet eredetileg mesterséges intelligencia kutatásához fejlesztette ki *John McCarthy* az egyesült államokbeli *Massachusetts Institute of Technology* nevű egyetemén még az ötvenes évek végén. A 80-as évek közepétől már, mint az AutoCAD alkalmazás fejlesztési felületeként (API) is számon tartják. A Visual LISP (röviden VLISP) tulajdonképpen az AutoLISP programok fejlesztésének megkönnyítésére készült az ehhez szükséges integrált fejlesztői környezet (*Integrated Development Environment*, rövidebben IDE) megalkotásával. A VLISP IDE néhány fontosabb szolgáltatása:

- Szintaktika ellenőrzése.
- Alkalmazások lefordítása, mely növeli a futási sebességet.
- A forráskód nyomkövetése.
- Változók értékének futás közbeni megfigyelése.
- Konzol ablak a hatékonyság növelése érdekében.

A VLISP környezet külön saját ablakokkal, menükkel rendelkezik, de ezek használatához az AutoCAD-nek futnia kell. (A VLISP fejlesztői környezet eleve AutoCAD menüből indítható.)

A Visual LISP for Autodesk Map a VLISP nyelvnek, kifejezetten térinformatikai felhasználásra tervezett függvényekkel való kiegészítéseként jött létre.
A függvények három nagy témakör köré csoportosíthatóak:

- Adatokkal kapcsolatos függvények.
- Térképkészítéssel kapcsolatos függvények.
- Topológiákkal kapcsolatos függvények.

A három nagy csoport számos további alcsoportra bomlik, melyekről, s a hozzájuk tartozó függvényekről a [4]-ben találhatunk teljes leírást. Azt azonban meg kell jegyeznünk, hogy az ide tartozó függvények, majdnem lefedik az Autodesk Map teljes funkcionalitását, gyakorlatilag csak az Oracle térbeli adatainak (*Oracle Spatial Data*) használatához nem nyújtanak támogatást.

3.6.3. Programozható párbeszédpanelek

A párbeszédpanelek felépítését egy speciálisan erre a feladatra kifejlesztett nyelv, a DCL (*Dialogue Control Language*) segítségével írjuk le. Az egyébként egyszerű ASCII szöveges állományt dcl kiterjesztéssel kell elmenteni, de létrehozása, módosítása nem csak a hagyományos szövegszerkesztőkkel lehetséges, hanem igénybe vehetjük a VLISP fejlesztő környezetét is. Ez utóbbi esetben olyan kényelmi szolgáltatás is rendelkezésünkre áll, mint a párbeszédpanel előnézeti képének megtekintése.

A párbeszédpanelek építő elemeit mozaikoknak (*tile*) nevezzük, melyek attribútumai határozzák meg külső megjelenésüket és funkciójukat. Az attribútumok lehetséges fajtái mozaik típusonként változnak. A mozaikok sorokba és oszlopokba csoportosíthatóak, melyek ezen felül még egymásba is ágyazhatóak. A csoportosítási lehetőséggel egy hierarchikus fa struktúra építhető fel, mely pontosan leírja a párbeszédpanel külsejét. A leggyakrabban előforduló előre definiált mozaik típusok:

- Nyomógomb (*Button*)
- Szerkesztő doboz (*Edit box*)
- Lista doboz (*List box*)
- Legördülő lista (*Popup list*)
- Rádió gomb (*Radio button*)
- Csúszka (*Slider*)
- Jelölő négyzet (*Toggle*)

Dekorációs célokra hozták létre a kép (*Image*), a szöveg (*Text*) és a helykitöltő (*Spacer*) mozaikokat. A mozaikok csoportosításához az oszlop (C*olumn*) és sor (R*ow*) típusú tárolók használhatóak.

A Visual LISP függvények egy külön csoportja nyújt segítséget a párbeszédpanelek alkalmazásból történő vezérlésének megvalósításához. Így például függvények töltik be a párbeszédpanelek szerkezetét a dcl állományokból a memóriába, majd ezek távolítják el használat után; kezdőértékeket rendelnek a mozaikokhoz, felhasználói bevitelt fogadnak, lekérdezik a felhasználó által megváltoztatott mozaikok értékét, vagy akár különböző tevékenységek végrehajtását indíthatják el a felhasználói inputtól függően.

„Ne mondjátok nekem, hogy
ez nehéz probléma.
Ha nem lenne nehéz, nem
lenne probléma.”
Ferdinand Foch
(francia hadvezér)

4. Városi tömegközlekedés modellezése hálózat topológiával

Az ember hétköznapi élete gyakran szolgáltat olyan kérdéseket, problémákat, melyeket érdemes végiggondolni, vagy mi több megoldani. Naponta többször kell például eljutnunk lakóhelyünkön egyik helyről a másikra, mely megvalósítása alapvetően három módon történhet: gyalog, tömegközlekedési eszköz igénybe vételével vagy autóval. Mindhárom esetben lakóhelyünk útvonalai közül kell azokat kiválasztanunk, melyek a kívánt cél(ok)hoz elvezetnek minket. A döntésünk meghozatalához elsősorban az utak, utcák helyzetét kell jól ismernünk, vagyis helyhez kötött információt kell elemeznünk.

Ilyen jellegű probléma szemléltetésére lokális térinformációs rendszer építhető fel, mivel a térinformációs rendszerek helyhez kötött információk gyűjtésére, kezelésére, elemzésére és megjelenítésére szolgálnak [8], s ha az adatok viszonylag kis területre (például egy településre) koncentrálódnak, akkor úgynevezett lokális rendszert készítettünk.

A most megalkotásra váró rendszer szempontjából a gyalogos közlekedés, tömegközlekedési eszköz használatával való kombinálása élvez kitüntetett szerepet, a következő indokok alapján:

- A tömegközlekedési eszközök útvonala előre rögzített, és csak diszkrét pontok között lehetséges az igénybe vétele.
- Gyalogosan a gyalogos forgalom számára engedélyezett utcák használatával tetszőleges pont megközelíthető.
- Autó használatával csak akkor nem érhető el egy pont, ha ott nem engedélyezett, esetleg fizikailag lehetetlen a gépjármű forgalom, vagy ha megállási nehézségek vannak. Ezekben az esetekben is elmondható azonban, hogy az úti cél egészen jól

megközelíthető volt. (Hazánkban még egyáltalán nem jellemző a városszéli nagy parkolók jelenléte, majd a tömegközlekedés használata.)

- A 3. fejezetben röviden ismertetett Autodesk Map szoftver szolgáltatásainak igénybe vételével csak egy teljes hálózatra vonatkozó elemzések végezhetőek el. Az egyes tömegközlekedési eszközök választhatóságának következtében viszont az elemző művelet hatáskörébe tartozó objektumok köre a beállításoktól függően változik.
- Egy cél eléréséhez átszállásra is szükségünk lehet, amit nem csak az aktuális megállóban lehet megvalósítani, hanem a közelben lévő, esetleg más típusú jármű meg megállóját is számításba kell venni.

Az elérendő célállomások körének meghatározása sok szempont alapján történhet, engem ebben személy szerint a bevezetőben említett munkahelyi kollégák (nem térinformatikai) fejlesztései befolyásoltak. Ennek eredményeképpen Debrecen városban található – elsősorban idegenforgalmi jelentőséggel bíró – épületek megközelítésének optimalizálásával foglalkozom a helyi tömegközlekedés használatával. A modell szempontjából az épületek funkciója természetesen nem bír különösebb jelentőséggel, bármikor kicserélhető lenne közintézmények, üzletek, szállodák, éttermek vagy bármilyen tetszőleges rendeltetésű épületek csoportjára.

A modell gyakorlati megvalósítását egy Visual LISP nyelven kifejlesztett Autodesk Map alkalmazás szemlélteti, mellyel szemben a következő elvárásokat támasztottam:

- Listából, minél több információ birtokában választhassa ki a felhasználó a meglátogatni kívánt helyszíneket (épületeket).
- Tömegközlekedési eszköz használata engedélyezhető legyen, az eszköz típusa ugyanakkor szabadon változtatható legyen.
- A leggyorsabban megtehető útvonalat határozza meg.
- Az előző útvonal beállításai módosíthatóak legyenek.
- Térképrészletet lehessen nyomtatni az ajánlott útvonalról.
- Szöveges riportot legyen készíthető az ajánlott útvonalról.

Az alkalmazás tervezésénél abból indultam ki, hogy az egy fix helyen működő számítógépre van telepítve (például szálloda, vasútállomás, turista információ), ezért nem kell minden útvonal meghatározásához a kiindulási pontot külön megadni. Ez az alapértelmezett indulási hely ennek ellenére természetesen megváltoztatható, elősegítve ezzel a szélesebb körű felhasználhatóságot.

4.1. Az elméleti modell megalkotása

A továbbiakban a valós világból (ez most Debrecen város) az utcák, és a tömegközlekedési eszközök útvonalai lesznek fontosak, így ezek alkotják majd az elméleti modell legfontosabb alapelemeit. Az entitások kétféle osztálya különböztethető meg aszerint, hogy hogyan jutunk el egyik pontból a másikba: gyalog, vagy tömegközlekedési eszközzel. (A lehetséges eszközök típusától – busz, villamos, troli – az entitások kezelése nem változik.)
Ezen entitások attribútumai a következők lehetnek:

- A haladás nyomvonalának utcaneve.
- A haladás lehetséges irányai.
- A haladás sebessége.
- Az igénybe vett jármű, típuson belüli azonosítója (járatok számozása).

Kiemelt jelentőséggel bírnak azok a pontok is, ahol a tömegközlekedési eszközök igénybevétele megkezdődhet, vagy befejeződhet, ezért megálló elnevezéssel entitásként bővítik a modellt. A megálló elnevezése attribútumként megadható.

Szintén entitásként definiáljuk az épületeket, melynek legfontosabb attribútumai az elnevezése és a megközelíthetőség szempontjából fontos bejárat elhelyezkedése.

Az entitások jellemzésére az osztályba sorolás és az attribútumok megadása mellett a közöttük lévő kapcsolatok meghatározása szolgál, melyeknek egy része elemzés során derül ki, míg másik részét attribútumként kell tárolni. A modellünk entitásai között értelmezhető kapcsolatokról a 3. táblázat ad áttekintést. (A táblázatban használt rövidítések: TK: tömegközlekedés, M: megálló)

	gyalog út	TK	Megálló	Épület
gyalog út	egymáshoz kapcsolódik		megállóhoz vezető	épület bejáratához vezető út
TK		azonos megálló (kezdő vagy vég)	M-ból indul M-ba érkezik	
Megálló	megállóhoz vezető	M-ból indul M-ba érkezik	szomszédos	épülethez legközelebbi megálló
Épület	épület bejáratához vezető		épülethez legközelebbi megálló	

3. Táblázat Az entitások között értelmezhető kapcsolatok

A modellalkotás következő lépése a kiválasztott entitások számítógépes reprezentációját biztosító adatmodell létrehozása, melyet az Autodesk Map rendszer jellemzőinek figyelembe vételével kell létrehoznunk.

4.2. Az adatmodell kiépítése Autodesk Map 6 környezetben

Az adatmodell megvalósításához három fő feladatot kell megoldani, a fejlesztési környezet eszközeinek felhasználásával. Először is gondoskodnunk kell minden entitás típus digitális reprezentációjáról, majd a szükséges attribútumok tárolását kell megoldani, végül a kapcsolatokat is realizálni kell.

Az útvonal elemzési feladatokat megvalósító Visual LISP programot úgy építettem fel, hogy ha bárki a fejezet útmutatásait pontosan követve megalkot egy tetszőleges települést reprezentáló adatbázist, kicserélve a jelenlegivel az alkalmazás továbbra is működni fog. Ez indokolja az igen részletes leírásokat, mely olykor még a fóliák elnevezésére is kiterjed.

4.2.1. Objektumok definiálása

A gyalogosan bejárható utcákat reprezentáló vonallánc objektumokat úgy kell létrehoznunk, hogy abból a szükséges elemzések elvégzésére alkalmas hálózat topológia kialakítható legyen. Ehhez vagy már az

objektumok kialakításánál megtörjük az egymást metsző objektumokat, vagy a rajz tisztázása során végeztetjük el ezt a munkát a rendszerrel. Az objektumok tárolására külön fóliát (Utca_hálózat) hozunk létre a forrásrajzban.

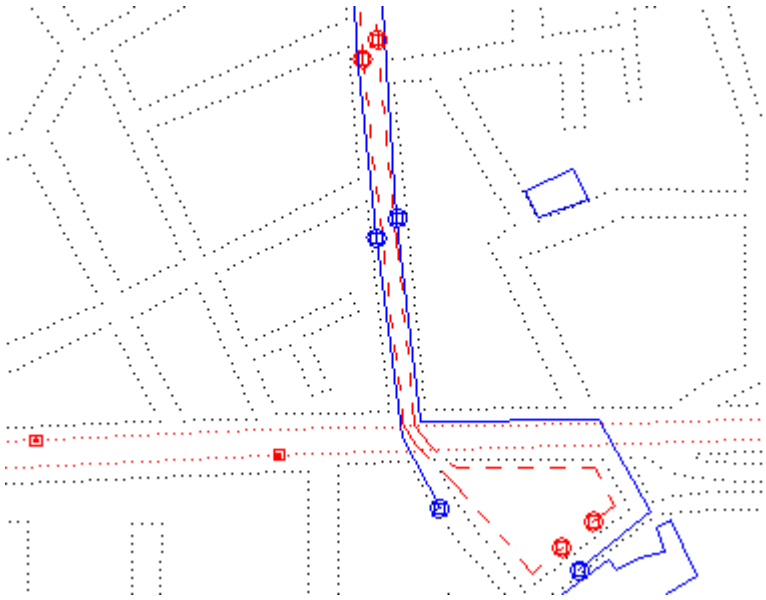
A közlekedési eszközök útvonalának önmagában való megvalósítása hasonlóan történik, mint az utcahálózaté. Eszköz típusonként külön fóliát hozunk létre a tárolásukhoz. A szükséges vonalláncok azonban csak bizonyos diszkrét pontok (a megállók) között haladhatnak, ezért előbb ezek létrehozását kell áttekintenünk.

8. Ábra Részlet az Utca_hálózat fólia objektumaiból

A megálló elnevezéssel illetett entitások számára blokk objektumot definiálunk, mely egy ACAD pont objektumból, és egy köré rajzolt négyzetből áll. A blokkot a 0 fólián kell definiálnunk, mert ekkor a blokk aktuális fóliára beillesztett példányai ténylegesen az aktuális fóliára kerülnek. Az egyes tömegközlekedési eszközök számára a saját fóliáikon (TK_busz, TK_troli, TK_villamos) a megállók elhelyezkedését figyelembe véve kell a blokkokat beilleszteni. E pontszerű objektumot azért kényszerülünk blokként ábrázolni, mert külső megjelenésében meg szeretnénk különböztetni további pont objektumoktól (például a csomópontoktól). Az Autodesk Map viszont egy rajzon csak egyféle pont stílust tud egyszerre használni.

A fóliák elnevezéseinek megválasztásában a helyettesítő karakterek felhasználhatósága vezérelt. Így például a tömegközlekedési eszközök útvonalait tartalmazó fóliákra történő hivatkozás egyetlen kifejezéssel megadható a *TK_** karaktersorozat segítségével.

A blokkok elnevezése, s tulajdonképpen a felépítése is szabadon megváltoztatható. Ezeket az információkat maga az alkalmazás nem használja, csak a topológia (4.2.3. fejezetben részletesen ismertetett) definiálása során van szükségünk az elnevezéseikre azonosításuk céljából.

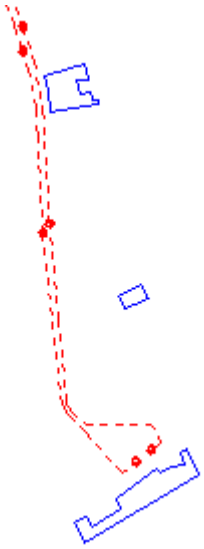


9. Ábra A három tömegközlekedési eszköz útvonalának és megállóinak megjelenítése

A megállók segítségével már létrehozhatóak a megfelelő fóliákra az érintett tömegközlekedési eszközök útvonalai két-két szomszédos megálló között. A megvalósításhoz újra vonallánc típusú objektumokat definiálunk a következő megkötésekkel:

- A tömegközlekedési eszközök útvonalai irányítottsággal rendelkeznek, ezért már az őket megjelenítő vonalláncok kezdőpontjának a kiválasztásakor vegyük figyelembe a hálózat topológiában kialakítandó irányt.

- A szomszédos megállók között abban az esetben is csak egyetlen vonalláncot kell definiálni, ha közöttük több, azonos típusú tömegközlekedési eszköz is jár teljesen megegyező útvonalon. (Például a Nagyállomás és a Petőfi tér között a 31-es, 32-es, 10-es, sőt a 14-es busz is közlekedik, de mivel az útvonaluk teljesen egybeesik, csak egyetlen vonalláncra van szükségünk.)
- Két, egymást követő megálló között haladó vonalláncot semmilyen körülmények között ne törjünk meg. (Az utcahálózattal való metszéspontok nem valódiak, hisz a jármű pályája kötött.)



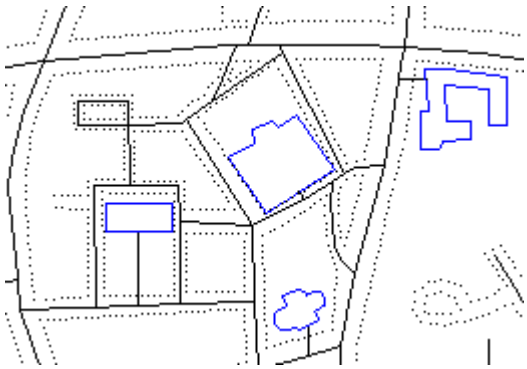
10. Ábra Épületek körvonalai és a villamos pálya nyomvonala

Az épületek ábrázolása az alaprajzot vázlatosan követő zárt vonallánc objektumok (látvány tekintetében poligonok) létrehozásával könnyen megoldható. Ezen objektumok tárolására külön fóliát (*Épületek*) kell létrehozni a forrásrajzban. Érdemes megfigyelni, hogy a zárt vonallánc adatait leíró asszociációs lista az egyszerű vonalláncéhoz képest valójában csak egyetlen tagban különbözik: a 70-es csoportkódot nyílt vonallánc esetén 0 követi, míg zárt vonalláncnál 1.

((-1 . <Entity name: 40097a90>) (0 . "LWPOLYLINE") (330 . <Entity name: 40084cf8>) (5 . "32A2") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "Épületek") (370 . 20) (100 . "AcDbPolyline") (90 . 4) ***(70 . 0)*** (43 . 0.0) (38 . 0.0) (39 . 1.0) (10 33.4561 24.0101) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 33.752 24.8233) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 34.4918 23.6158) (40 . 0.0) (41 . 0.0) (42 . 0.0) (210 0.0 0.0 1.0))

((-1 . <Entity name: 40097a90>) (0 . "LWPOLYLINE") (330 . <Entity name: 40084cf8>) (5 . "32A2") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "Épületek") (370 . 20) (100 . "AcDbPolyline") (90 . 4) ***(70 . 1)*** (43 . 0.0) (38 . 0.0) (39 . 1.0) (10 33.4561 24.0101) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 33.752 24.8233) (40 . 0.0) (41 . 0.0) (42 . 0.0) (10 34.4918 23.6158) (40 . 0.0) (41 . 0.0) (42 . 0.0) (210 0.0 0.0 1.0))

Az épületek bejáratának jelzéséhez a megfelelő helyen a poligon belsejéből kiindulva adjunk meg egy vonalláncot, mely az előtte elhaladó utcába csatlakozik (ha szükséges annak megtörésével). Ezeket az objektumokat is külön fóliára (*Hoz*) gyűjtjük, melyből eredő előnyöket később jól fel tudjuk használni.



11. Ábra Az épületek csatlakozása az utcákhoz

A tömegközlekedést megvalósító járművek útvonalának kötöttsége miatt a leírásukhoz nem volt használható a már létrehozott utcahálózat, hanem kénytelenek voltunk egy gyakorlatilag független objektum halmazzal megadni a különböző járatok pályáját. Önmagában tekintve ezen objektumokat megállapíthatjuk, hogy hálózat építhető fel belőlük, azaz az

elemző műveletek alkalmazhatóak rajtuk. Probléma viszont, hogy nem tudjuk tetszőleges helyről indítani az útvonalkereséseket, és az épületeket is csak a valamilyen értelemben legközelebbinek titulált megállóig tudjuk megközelíteni, így összességében a kitűzött feladatnak még nem tudunk eleget tenni.

Elkerülhetetlen tehát, hogy egyetlen hálózat topológiába foglaljuk mind az utcákat mind a járművek útvonalait reprezentáló objektumokat, de úgy, hogy az elemzésekhez szükséges résztopológiák később kialakíthatóak legyenek. A résztopológia fogalmát és alkalmazhatóságát a 4.5.1. fejezetben ismertetem részletesen, egyelőre elegendő arra koncentrálnunk, hogy kapcsolatot kell teremteni az utcák és a megállók között.

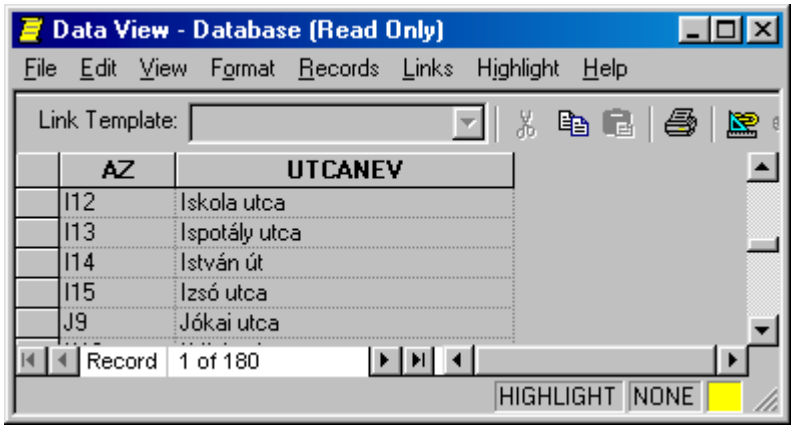
Az új objektumok fóliája *At* elnevezéssel kerül be a forrásrajzba, mely az átszállás szóra utal. Az utcahálózat és a megállók közötti hiányzó kapcsolatotokat vonalláncok valósítják meg, melyek segítségével a már meglévő objektumok egyetlen nagyobb méretű hálózattá alakíthatóak, lehetővé téve ezáltal az egységes kezelésüket. A vonalláncok létrehozásának iránya nem fontos ebben az esetben, hiszen a megállók és az utcahálózat élei között a lehetséges haladási irány tetszőleges lehet.

4.2.2. Az attribútumok megadása

Az entitások tulajdonságainak az objektumok tulajdonságai felelnek meg, csak ez utóbbiak a számítógépek számára feldolgozható módon írják le az attribútumokat. Az Autodesk Map környezetben rendelkezésünkre álló lehetőségeket a 3.2. fejezet ismerteti, míg a modellünkben fontos tulajdonságok a 4.1. fejezetben találhatóak.

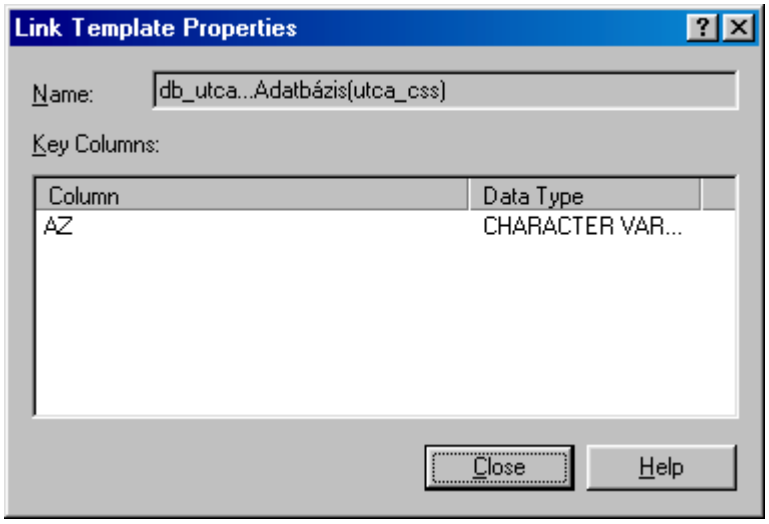
Az utcahálózatot alkotó éleknek az elnevezései (gyakorlatilag az utcanevek) egy külső adatbázisban kerülnek tárolásra az alábbi okokból kifolyólag:

- Előnyös lehet, ha az utcanevek listájához más, külső alkalmazás is hozzáférhet.
- Az esetleges változások rögzítése kényelmesebb és gyorsabb, hiszen csak egy helyen kell a változtatást elvégezni. (Mint tudjuk az utcák elnevezései változhatnak, sőt időnként új utcák keletkeznek.)
- Az Adatnézet ablak (10. ábra) funkciói jól használhatóak.



12. Ábra Az Adatnézet ablak

Figyelni kell viszont arra, hogy a csatolási sablon kulcsának definiálásában az utcanevet tartalmazó mező nem szerepelhet, mivel az így keletkező csatolási adat nem tenné lehetővé az adatbázis konzisztens frissítési lehetőségét (lásd 3.2.2. fejezet). Az utcák elnevezéseit tartalmazó külső adatbázis típusa lényegtelen, ha annak szerkezete (mezőinek elnevezése) a jelenlegivel megegyező.



13. Ábra A csatolási sablon tulajdonságai

Kifejezetten fontos még, hogy a csatolási sablon elnevezése (utca_css) ne változzon, mert arra explicit hivatkozások találhatóak az alkalmazásban (például SQL típusú lekérdezésekben, kifejezésekben).

A meglátogatandó épülete(ke)t egy nagyobb csoportból kell kiválasztanunk, amit egy felhasználóbarát alkalmazásnak illik minél jobban elősegíteni. Ez viszont azt jelenti, hogy érdemes az épületeket minél több attribútum felhasználásával jellemezni, megkönnyítve ezzel a választást. Mielőtt rátérnénk azonban ezek megvalósításának problémájára, meg kell említenem az eredményeimet bemutató alkalmazás egy másik szolgáltatását, mely az attribútumok tárolásának módját jelentősen befolyásolta.

14. Ábra A nyelv kiválasztását megvalósító

Tekintettel az alkalmazás felhasználási lehetőségeinek növelésére, az adatbázisokat, és a programkódot úgy készítettem el, hogy az, több nyelven is képes legyen kommunikálni a felhasználóval. Jelenleg az alkalmazás magyar mellett angol nyelven is használható, de további nyelvek használatára is könnyen megtanítható. (Erről bővebben a 4.7.2. fejezetben írok.)

Az adatbázisok tervezésének szempontjából a nyelvfüggő adatokat (elnevezés, leírás) több nyelven kell tárolni úgy, hogy egyszerre csak az alkalmazás aktuális nyelvén tárolt adatok legyenek elérhetőek. Az ilyen problémák legkézenfekvőbb megoldása az adatok nyelvenként külön strukturált txt állományban való tárolása, melyből az aktuális nyelvre vonatkozó adatokat az alkalmazásból listába beolvasva programváltozóként érjük el. (Objektumadat tábla használata fölöslegesen növelné a rajz méretét, s külső adatbázis használatával is nehezebben valósítható meg, hogy az éppen nem használt nyelv adatai ne terheljék az erőforrásainkat.)
Az épületekre vonatkozó attribútumok:

- Megnevezése
  Az épület elnevezése, melyet a fentebb említett okokból txt állományban tárolunk. Az épületek azonosítására viszont

szükségünk van egy, az aktuális nyelvtől független azonosítóra, melyet objektumadatként az *epulet* nevű tábla *azon* mezőjében tárolunk. (Azonosítóként mindig egy nagybetűs 5 karakterből álló, az épületre utaló rövidítést használunk. Például: Aranybika Szálloda –> ARANY, Kistemplom –>KISTM, Református Főiskola –> REFOI)

- Rövid leírás
  Az idegenforgalmi látványosságról (épület) rövid szöveges állomány. Az állományok neve három részből épül fel: az épület azonosítója + aláhúzásjel + nyelvet azonosító kettő hosszúságú karakterlánc (például: arany_hu.txt). A szöveges állományok első sora a megnevezést tartalmazza, míg a második sorban a leírás adható meg.
- Kép
  Az épületről készült fotó jpg, vagy más, a Microsoft Photo Editor által ismert fájlformátumban. Ez az attribútum nem függ az alkalmazás aktuális nyelvétől, ezért objektumonként csak egyetlen állományra van szükség.
- Legközelebbi csomópont azonosítója
  Az utca nevű hálózat topológia, az épület bejáratához legközelebb eső csomópontjának azonosítója. Tárolása az épület objektumok elnevezését is tartalmazó szöveges állományban történik a 4.2.4. fejezetben megfogalmazottak figyelembe vételével.

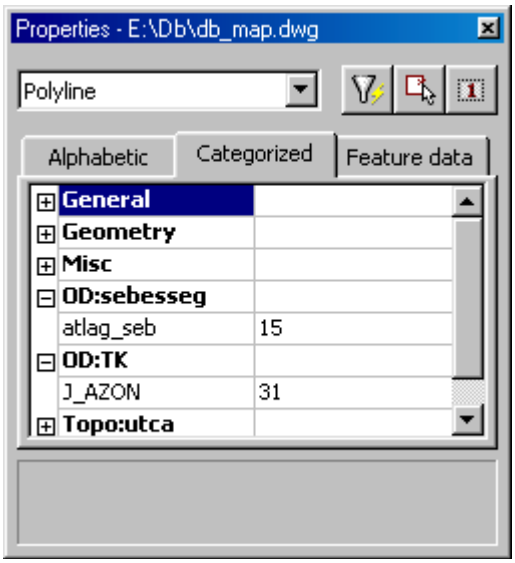
Az említett objektumjellemzőket tartalmazó állományok mappákban való elhelyezéséről – a könnyebb áttekinthetőség végett – a 4. táblázat nyújt információt. (Az állomány típusokat a tömörség kedvéért egy-egy példa segítségével adom meg. A harmadik oszlopban feltüntetett mappák mindegyike a meghajtó álnév (lásd 3.1. fejezet) által meghatározott mappából nyílik.)

A megállók egyetlen jellemzőjét objektumadat táblában tartjuk nyilván, melyből elegendő egyetlen, hisz a tömegközlekedési eszköz típusától független fogalom a megálló elnevezése. A létrehozandó tábla neve *Stop*, egyetlen karakteres típusú mezője *Neve* címkével definiálandó, rekordjai manuálisan csatolandóak.

Állomány típus	Leírása	Mappa
latniv_hu.txt	Épület objektumok elnevezését, és a csomópont azonosítót tartalmazó állomány (nyelvenként egy).	LATNI
arany_hu.txt	Épület objektum leírását tartalmazó állomány (épületenként, és nyelvekként egy).	LATNI
arany.jpg	Épület objektumok fotóját tartalmazó állomány (épületekként egy).	KEPEK

4. Táblázat Attribútumokat tároló állományokat tartalmazó mappák

A tömegközlekedési eszközök járatainak útvonalát meghatározó élekkel kapcsolatosan többféle jellemzőt tárolunk. A hálózat topológia felépítésekor a rendszer minden él objektumot kétirányúnak definiál, ami jelen esetben a tömegközlekedési eszközök útvonalának tekintetében nem fedi a valóságot.



15. Ábra Tömegközlekedési eszköz attribútumai

Ha az él objektumok definiálásakor a 4.2.1. fejezet utasításait betartottuk, akkor minden a TK_* fólián elhelyezkedő él előre mutató iránnyal

rendelkezik. Ebben az esetben TPMLINK_utca táblában az irány mező értékét kell 0-ról 1-re változtatni. E művelet manuálisan kicsit időigényes, ezért erre a célra írtam egy *Feltolt* nevű függvényt, mely meggyorsítja a munkánkat, s melynek leírása még ebben a fejezetben megtalálható.

Szükségünk van továbbá a szomszédos megállók között haladó tömegközlekedési eszközök járatszámának tárolására is. Ezt az adatot egy új objektumadat táblában tartjuk nyilván (15. ábra). A háromféle eszköz számára elegendő egyetlen táblát (TK) megadunk, egyetlen karakteres mezővel (J_AZON), hiszen a járatok azonosítására szolgáló karakterláncok nem csak kizárólag számjegyeket tartalmazhatnak.

A hálózatelemző funkciókat nem csak a legrövidebb, hanem a leggyorsabban megtehető útvonal meghatározására is fel tudjuk használni. Az ehhez szükséges kifejezés megalkotásához, az egyes éleken való haladás sebességére is szükségünk van. Ezt a sebesség adatot célszerű a rajzban tárolni, mert a hálózatot alkotó minden élnek szüksége van rá.

Az objektumadat táblát *Sebesseg* elnevezéssel, egy – akár alapértelmezett értéket is tartalmazó – numerikus mezővel (*atlag_seb*) hozzuk létre. Az egyes eszközökhöz rendelt sebesség értékek a valóság függvényében tetszőlegesek lehetnek, sőt még azonos jármű típus és járatszám esetén sem muszáj konstans értékeket megadnunk. (Ha valamelyik utcában forgalomkorlátozás van érvényben, akkor ez a táblában is könnyen rögzíthető.) Az adatok feltöltésének megkönnyítésére jól használható a CD-n megtalálható *rutin.lsp* állományban definiált következő függvény.

(feltolt *folia_nev tabla mezo rsz ertek*)

A paraméterek jelentése:

folia_nev	A fólia elnevezése, melyeken található vonalláncok objektumadatai változnak.
tabla	A megváltoztatandó adatokat tároló tábla neve.
mezo	A megváltoztatandó adatot tároló mező elnevezése.
rsz	Az objektumhoz csatolt mező indexe. (Nullával kezdődik !!!)
ertek	Az előbbi mezőben rögzítendő új érték.

Visszatérési értéke: True

Adatállomány neve és mezői	Attribútum tárolási módja	Mely objektumokról tárol adatot
kulso_db.dbf	Külső adatbázis	A hálózat topológia összes Utca_halozat fólián található objektumáról.
Stop • neve	Objektumadat tábla	Megállókról
Epulet • azon	Objektumadat tábla	Épületekről, és a Hoz fólia épületek bejáratához vezető objektumairól.
Sebesseg • atlag_seb	Objektumadat tábla	A hálózat topológia minden él objektumáról.
Tk • j_azon	Objektumadat tábla	A hálózat topológia minden TK_* fólián lévő él objektumáról.
latniv_hu	Txt állomány	Épületekről.
epuletazonosito.jpg	Jpg állomány	Épületekről
*epuletazonosito*_hu	Txt állomány	Épületekről.

5. Táblázat Attribútumok összefoglaló táblázata

Ahhoz, hogy az időmennyiséget meghatározó számításaink minél jobban megközelítsék a valóságot, az átszállásokat súlyozni szükséges. Ez legegyszerűbben az *At* nevű fólián található objektumok *TPMLINK_utca* táblában tárolt direkt és fordított ellenállás értékeinek módosításával oldható meg.

Az entitás típusokról összességében igen sok attribútumot tároltunk, ezért az eligazodás megkönnyítéséhez az 5. táblázat foglalja össze a fejezetben található formális információkat. (A dőlt betűs szavak a jelentésüknek megfelelő konkrét értékekkel helyettesítendőek.)

4.2.3. A hálózat topológia definiálása

Ha a hálózat topológia felépítéséhez szükséges valamennyi csomópont és él objektumokat elkészítettük, és letisztáztuk a rajzot, akkor a Map menü Topology almenüjének Create parancsával elkezdhető a

topológia definiálásához szükséges adatok megadása. Négy párbeszédpanelt kell kitöltenünk tetszőleges sorrendben az alábbi útmutatás alapján.

Az első párbeszédpanelen a topológia típusának válasszuk a Network rádiógombot, majd elnevezésnek Utca –t írjunk be. Az elnevezés fontos, hisz az alkalmazásban található különböző műveletek név szerint hivatkoznak rá. A leírásnak tetszőleges karaktersorozat adható meg.

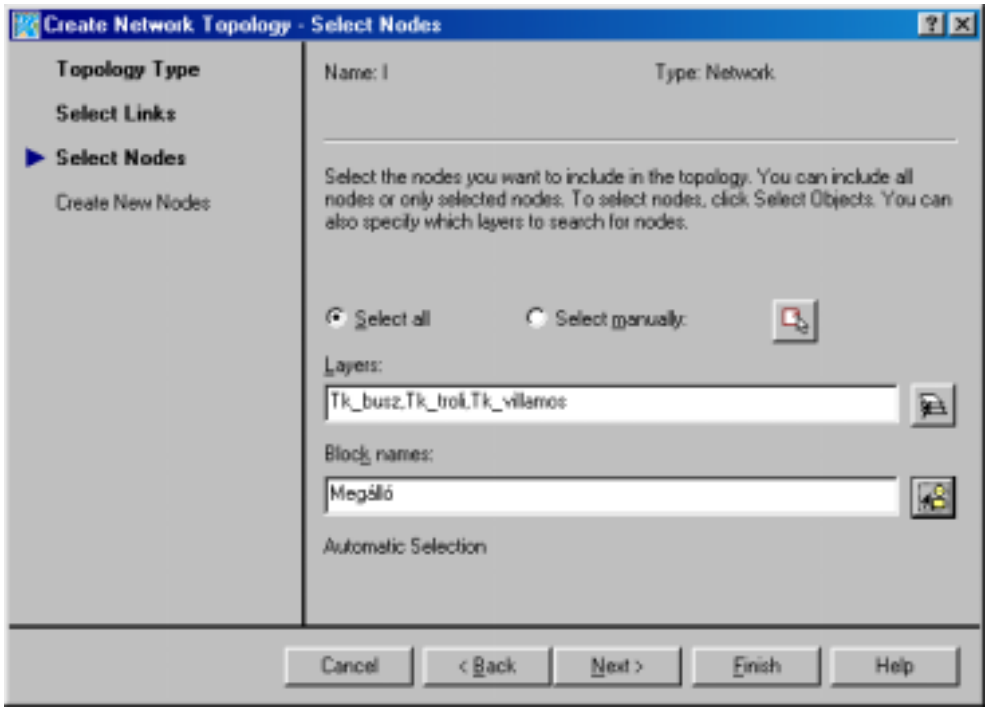
A második párbeszédpanelen a hálózatot alkotó élek fóliáit kell kiválasztani a  nyomógombra megjelenő listából, s a Select all rádiógombot kell kijelölni. Az éleket tartalmazó fóliák: At, Hoz, TK_busz, TK_villamos, TK_troli, Utca_hálózat.

A harmadik párbeszédpanelen a topológia létrehozásához csomópontként felhasználható, már létező pontszerű objektumokat tartalmazó fóliákat, és az objektumokat megvalósító blokk elnevezéseket kell megadnunk. A szükséges beállítások a 16. ábrán láthatóak.

Végül az újonnan keletkező csomópontok számára kell pont objektum típust (ACAD_POINT) és fóliát (Cspontok) választani. A beállítások végrehajtásához előbb természetesen a jelölő négyzet segítségével engedélyezni kell új csomópontok készítését. Új csomópontokra azért van szükségünk, mert a vonalláncok végződésein nincsenek pont objektumok.

A Finish nyomógombbal elindíthatjuk a topológia készítését, mely az új csomópont objektumok létrehozásán felül gyakorlatilag négy, önálló objektumadat tábla definiálását és azok, objektumokhoz csatolt adatokkal való feltöltését jelenti.

- TPMDESC_utca
  A topológia definiálásához szükséges adatok részletes leírása, melyeket például a topológia újból történő létrehozásakor használ fel a rendszer.
- TPMID_utca
  Az objektumok azonosítására kiosztott utolsó azonosító értéke.
- TPMLINK_utca
  Az él objektumok topológiai adatai.
- TPMNODE_utca
  A csomópont objektumok topológiai adatai.



16. Ábra Csomópontok meghatározása

Az újonnan keletkezett táblákban még el kell végeznünk a 4.2.2. fejezetben ismertetett változtatásokat, annak érdekében, hogy az adatok valósághűen írják le a hálózatot. Ennek megvalósításához szintén az említett fejezetben bemutatott *feltolt* függvény nyújt segítséget.

4.2.4. Kapcsolatok felderítése

A 3. táblázatban összefoglalt kapcsolatokat digitális formában is rögzíteni kell. A 4.2.2. fejezetben tárgyalt attribútumok létrehozásával, illetve az *utca* topológia 4.2.3. fejezetben bemutatott definiálásával a kapcsolatok egy jelentős részét már rögzítettük is. A hálózatelemző műveletek használatához a kezdő és közbenső (csomó)pontok topológiabeli azonosítójának gyors elérésére lesz szükségünk. Ehhez – mint már említettük – egy olyan szöveges állományt készítünk (*latniv_hu.txt*), melynek minden sora egy-egy épületről tartalmazza a következő információkat, egyetlen (!) helyközzel elválasztva:

- épület 5 betüs azonosítója,
- 4 hosszan a csomópont azonosító,
- épület megnevezése

A 17. ábra egy ilyen állomány tartalmából mutat be néhány sort.

```
ARANY 0605 Arany Bika Szálloda
DEKLT 2109 Debreceni Egyetem (TTK BTK)
DEOEC 2111 DE Orvost. Centrum
DERIM 0532 Déri Múzeum
IPARK 2107 Kereskedelmi és Iparkamara
KISTM 0664 Kistemplom
MEGYE 2103 Megyeháza
ALLOM 2105 Nagyállomás
NAGYT 0559 Nagytemplom
REFII 0525 Református Kollégium
REFOI 0497 Ref. Tanítóképző Főiskola
SZINH 0630 Színház
VAROS 0654 Városháza (régi)
```

17. Ábra A latniv_hu.txt állomány kivonatos tartalma

A legközelebbi csomópont meghatározása több módon történhet. Az 5. fejezetben ismertetett *pont a poligonban* algoritmus egy korábbi eredményem, mely akár a legközelebbi csomópont felderítésére is felhasználható, ha az alkalmazásának előfeltételeit megteremtjük. Azt hiszem magától értetődő, hogy ebben az esetben kár lett volna kihagynom az algoritmus gyakorlatban való tesztelését, még akkor is, ha jelentős külön munkát jelentett az algoritmus Visual LISP nyelven való implementálásának elkészítése.

A 4.2.1. fejezetben a *Hoz* fólia objektumainak leírásakor már az előbbi szempont vezérelt, amikor az objektumokat reprezentáló vonalláncokkal kapcsolatosan azt a kikötést tettem, hogy a vonallánc induljon a poligon belsejéből. Az épületekhez legközelebbi csomópont ezek után nem más, mint az *Hoz* fóliának az épület azonosítójával megegyező azonosítóval rendelkező objektumának az épületet jelképező poligonon (zárt vonalláncon) belül elhelyezkedő végpontja.

A forrásrajzon futtatandó *latniv_keszit* függvény az Épületek fólián található objektumokhoz keres legközelebbi, a hálózathoz tartozó csomópontot.

Eredményképpen két állomány készül(het):

- *latniv.txt*
  A 10. ábrán bemutatott szerkezetű szöveges állomány, mely nem tartalmazza az épületek elnevezését.
- *hiba.txt*
  Az épületek azonosítóját és a hibaüzenetet tartalmazza azokról az objektumokról, amelyeknél valamilyen okból nem járt sikerrel a függvény. Nem kerül létrehozásra, amennyiben nem lépett fel hiba. (Az állományban olvasható hibaüzenetek is az aktuális nyelven jelenik meg.)

A *latniv.txt* állomány mintegy sablonként használható a továbbiakban: a sorokat kiegészítjük a kívánt nyelven az épületek elnevezésével, majd más néven elmentjük a már ismertetett szabályok szerint (például *latniv_hu.txt*.) Az így előállított fájl (17. ábra) beolvasása után a lista típusú programváltozó tartalma a 18. ábrán látható. (A lista felépítését és tulajdonságait a 4.4. fejezet tárgyalja.)

```
(("ARANY" 2 "0605" "Arany Bika Szálloda") ("DEKLT" 0 "2109" "Debreceni Egyetem
(TTK BTK)") ("DEOEC" 0 "2111" "DE Orvost. Centrum") ("DERIM" 0 "0532" "Déri
Múzeum") ("IPARK" 0 "2107" "Keresk. és Iparkamara") ("KISTM" 0 "0664"
"Kistemplom") ("MEGYE" 0 "2103" "Megyeháza") ("ALLOM" 0 "2105" "Nagyállomás")
("NAGYT" 0 "0559" "Nagytemplom") ("REFII" 0 "0525" "Református Kollégium")
("REFOI" 0 "0497" "Ref. Tanítóképző Főiskola") ("SZINH" 0 "0630" "Színház")
("VAROS" 0 "0654" "Városháza (régi)"))
```

18. Ábra Részlet a listadobozok elemeit nyilvántartó lista tartalmából

4.3. A forrásrajz megvalósításának további kérdései

Az alkalmazás működéséhez szükséges grafikus adatok nyerésének számtalan különböző módja lehetséges, mely hatással van az adatminőségre is [8]. Az adatminőség meghatározása az alábbi négy, a megalkotandó térinformatikai rendszer szempontjából súlyozott tényező figyelembe vételével történik:

- A tényleges igények.
- A költségek.

- A megvalósíthatóság.
- A rendelkezésre álló idő.

Ugyanaz a feladat különböző minőségű adatokkal is megoldható az említett tényezők súlyának megválasztásától függően. Ebben a fejezetben az általam használt adatbázist az adatnyerési eljárás típusa alapján jellemzem, és kitérek az alkalmazott adatok minőségére is.

4.3.1. Digitális állományok átvétele

Világviszonylatban megfigyelhető tendencia, hogy egyre gyakrabban alkalmazzák a már meglévő digitális adatállományok átvételét. A nemzetközi és a nemzeti szabványosítási törekvéseknek köszönhetően ez a másodlagos adatnyerési eljárás egyre csak egyszerűsödik, ami tovább növeli népszerűségét.

A CD mellékleten található db_varos.dwg állomány egy, a *ForVill 2002 Kft*. által rendelkezésemre bocsátott digitális állomány feldolgozásaként állt elő. Az említett cég kizárólag a kutatásaim szemléltetésére kifejlesztett alkalmazás által történő hasznosítás céljából engedélyezte az általa készített digitális állomány átvételét. Jelen esetben ez az adatnyerési eljárás gyors, alacsony költségű, és a rendszer céljának megfelelő adatokat biztosított.

Nagyon előnyös volt, hogy az átvett adat formátumát nem kellett megváltoztatni. Az AutoCAD szabványos dwg rajzállománya tartalmazta a város közterületeinek, fontosabb épületeinek határvonalát, a vasút nyomvonalát illetve a patakok folyását. Mindez egyetlen fólián volt elhelyezve, amit én tematikusan több fóliára csoportosítottam felhasználva a fóliákhoz rendelhető sajátosságokat. Ezt követően a hálózat topológia definiálásához szükséges vonalas objektumok kialakítását, az attribútumadatok definiálását a korábbi fejezetekben ismertetett módon elvégeztem.

4.3.2. Az adatminőség

A rendszer célja – a szemléltetésen túl – útvonalterv készítése a felhasználó által meghatározott paraméterek alapján. A paraméterek körébe a helyváltoztatáshoz felhasználható tömegközlekedési eszközök típusa is beletartozik.

Az adatminőségre az alábbi tényezők gyakorolnak leginkább hatást [8]:

- Az adatok eredete.
- A geometriai pontosság.
- Az attribútum adatok tartalmi pontossága.
- A geometriai és az attribútum adatok konzisztenciája.
- A geometriai adatok (topológiai) konzisztenciája.
- Az adatok teljessége.
- Az adatok aktualitása.

A projektben felhasznált rajzfájl digitalizálással készült az említett cégnél egy nagy léptékű térképről. A munkafolyamatot igen nagy pontosság jellemezte, így a digitális állomány hűen tükrözi a papíralapú térkép objektumait.

Szerettem volna, ha egy EOV-ben készült térkép digitális változatát használhatom, de sajnos a rendelkezésemre álló idő alatt nem sikerült megfelelő minőségű változathoz hozzájutnom. Ugyanakkor el kell ismerni, hogy a rendszer céljának eléréséhez nem feltétlenül szükséges az említett vetületi rendszer használata, ezért egyelőre ettől eltekintettem. (Hazánkban 1972-ben került bevezetésre az *Egységes Országos Vetületi Rendszer*, mely egy ferdetengelyű, redukált, konform hengervetület, amelynek alapfelülete az IUGG/67 ellipszoid.[11])

Az átvett digitális állomány attribútumadatot nem tartalmazott, így csak azok definiálása során kellett a tartalmi pontosságra és konzisztencia biztosítására figyelni. Tartalmi pontatlanság felmerülése esetén helyszíni bejárással ellenőriztem az adatokat.

4.4. A párbeszédpanelt kiszolgáló adatstruktúrák és függvények

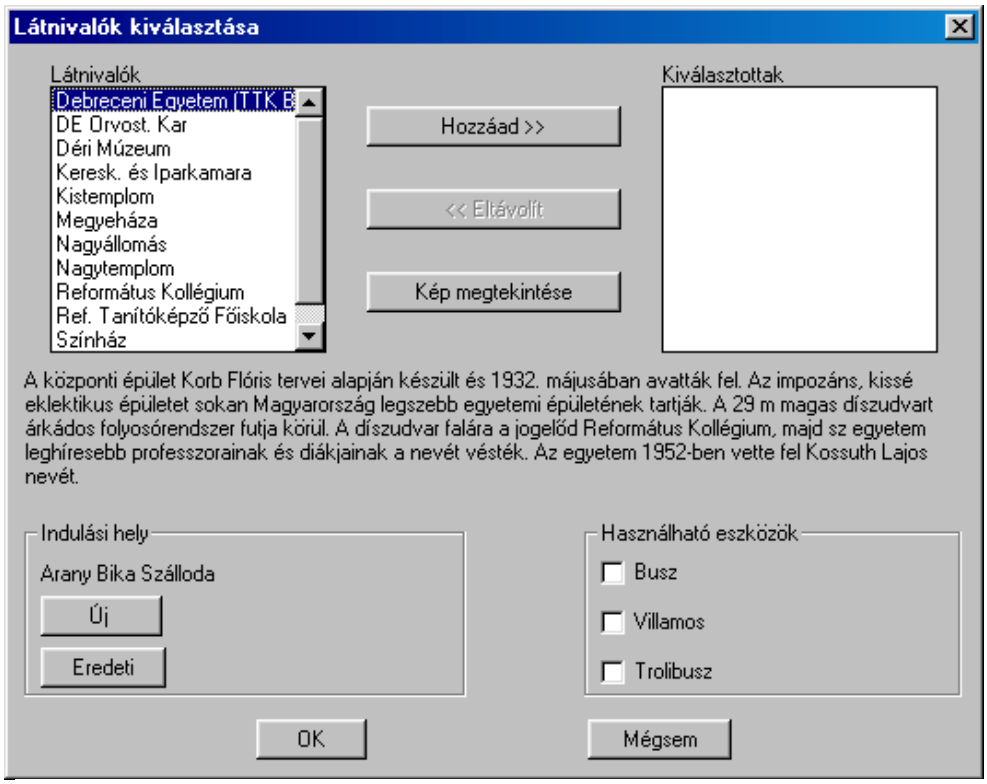
Az adatbázisok gyakorlatilag elő vannak készítve az elemző művelet végrehajtására, már csak a párbeszédpanelt kiszolgáló függvények, és programváltozók ismertetése van hátra. Azért térek ki részletesen erre, mert a két listát kezelő függvényt úgy építettem fel, hogy bármely, két lista dobozt kezelő alkalmazásban is hasznosítható legyen, kiküszöbölve a lista dobozok használata közben felmerülő néhány nehézséget:

- A lista dobozokban kijelölt elem lekérdezésekor csak az elem indexét (listában elfoglalt helyének sorszáma) adja vissza a lekérdező függvény.
- A lista dobozok tartalmának karbantartása közben, az elemek (jelen esetben betű szerinti) rendezettségének fenntartása problémás.

A kiolvasott adatokat (9. ábra) először is egyetlen, strukturált listává kell alakítani. (A programban *latni_lb* névvel illettem, ahol az lb a mozaik típusára utal: *list box*.). A lista betű szerinti rendezettségéhez elegendő a szöveges állomány épület elnevezések szerinti rendezettségének biztosítása. Tekintettel arra a korábbi megállapításomra, mely szerint egy elem helyváltoztatása csak egy jelölő érték megváltoztatását jelenti, megállapíthatjuk, hogy a listaelemek dobozok közötti mozgása nem befolyásolja az elemek fizikai sorrendjét. Ennek köszönhetően a listából, a jelölő értékek segítségével keletkező allisták is öröklik a betű szerinti rendezettséget.

A *lista_lb* elemei a szöveges állomány egy-egy sorából keletkeznek, a következő szabályok szerint.

- 5 betűs azonosító (az első öt karakterből)
- kiválasztottságot jelző numerikus érték
  0: nincs még kiválasztva
  1: ki van választva
  2: indulási hely
- 4 hosszan a csomópont azonosító (karakterek: 7-10)
- tetszőleges elnevezés (karakterek: 12-)



19. Ábra

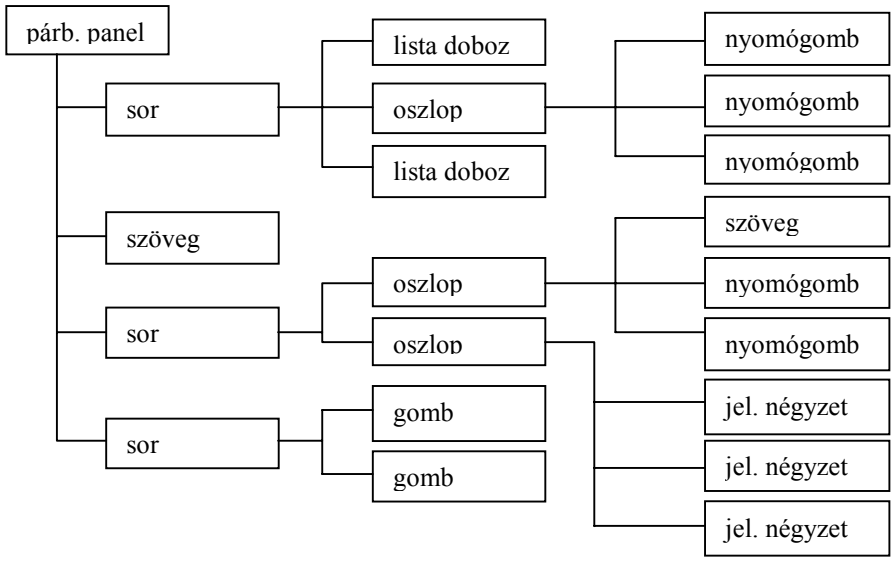
Az így kialakított szerkezet további előnyei:

- A listát alkotó elemek között könnyen lehet keresni az *assoc* listakezelő Visual LISP függvény és az 5 betűs azonosító segítségével.
- A numerikus érték segítségével a listadoboz tartalma könnyen meghatározható.
- A lista elemeinek karbantartására a Visual LISP *subst* függvénye kiválóan alkalmas.
- Tekintettel az elemek listabeli helyének változatlanságára, bármelyik listadoboz elemeinek rendezettsége automatikusan előáll.

Az elemek lista dobozok közötti mozgását az *Oda* illetve a *Vissza* elnevezésű függvények valósítják meg. Egyetlen paraméterükben a kiválasztott elem új helye adható meg. (Új indulási helynek is jelölhetünk egy elemet az Új nyomógombbal.) A két függvény működését az alábbi lépések írják le:

- Listadoboz kiválasztott elemének indexének lekérdezése.
- Az adott indexű elem programváltozóbeli indexének meghatározása.
- Jelző bit értékének megváltoztatása.
- Listadobozok újratöltése.
- További párbeszédpanel elemek frissítése.

A párbeszédpanel mozaikokból (*tile*) való strukturált felépítése a 20. ábrán látható, a megfelelő kódot pedig a mitnezmeg_hu.dcl állomány tartalmazza. (Az ábrán nem tüntettem fel a helykitöltő mozaikokat (*Spacer*), mert azok nem befolyásolják a párbeszédpanel alapvető funkcionalitását, csak a külső esztétikus megjelenésére vannak hatással, viszont kellőképpen rontanák az ábra áttekinthetőségét.)



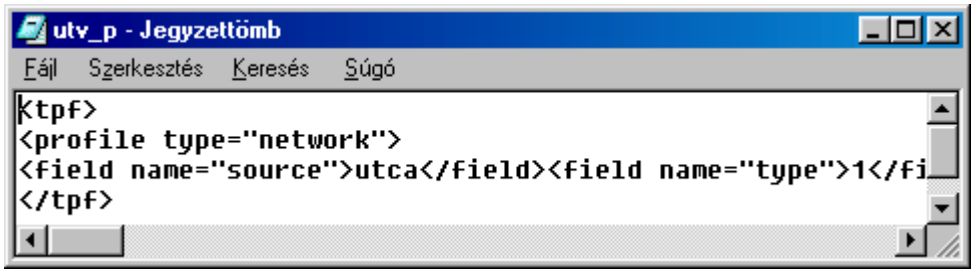
20. Ábra A párbeszédpanel mozaikokból való felépítése

4.5. Az adatelemzés rész-hálózat topológián

A hálózat topológián végrehajtható elemző műveleteket a 3.5. fejezetben már bemutattam. Az alkalmazás ezek közül a Legjobb út (*Best route*) elnevezésű műveletet használja, hiszen a célkitűzésemben egyetlen kiindulási pontból, tetszőleges úti célokat érintő, majd a kiindulási pontba visszatérő útvonal készítését határoztam meg.

A művelet pontos végrehajtását definiáló paraméterek egy úgynevezett profile állományba menthetőek el. A profile fájl tulajdonképpen egy szigorú szintaktikai szabályok szerint felépülő szöveges állomány, *tpf* kiterjesztéssel.
Az alábbi információk nyerhetőek ki belőle:

- A hálózat típusa.
- A topológia neve.
- Élek direkt ellenállását definiáló kifejezés.
- Élek irányát definiáló kifejezés.
- Csomópont ellenállását definiáló kifejezés.
- Él fordított ellenállását definiáló kifejezés.
- Az eredmény topológia elnevezése.
- Az eredmény topológia leírása.
- Az útvonal maximum ellenállása.
- Az útvonal minimum ellenállása.
- Megjelenítés színe
- Kezdő csomópont topológiabeli azonosítója
- Meglátogatandó csomópontok topológiabeli azonosítója



21. Ábra Részlet egy profile állomány tartalmából

A szintaktikai szabályok betartásával készített profile segítségével alkalmazásból is indítható az elemzés. A 19. ábrán látható párbeszédpanel OK nyomógombjára történő kattintást követően, a program elkészíti a profile állományt (forráskód *prof_ir* függvénye), majd betölti a megfelelő résztopológiát (lásd 4.5.1. fejezet), s végül az eltárolt paraméterek segítségével lefuttatja az elemzést, s megjeleníti a projektrajzon a végeredményt.

Az elemzéshez használt profile fájl teljes tartalmának megtekintéséhez, nyissuk meg tetszőleges szövegszerkesztővel az Autodesk Map 6 telepítési mappájában található, az alkalmazás által készített utv_p.tpf állományt.

4.5.1. Résztopológia fogalma és jellemzői

Mint láttuk, az elemző műveletek egyetlen topológiához tartozó objektumokon dolgoznak. Jelen esetben viszont a párbeszédpanelben található három jelölőnégyzet kitöltésétől függően az útvonal meghatározáshoz igénybe vehető élek halmaza változó, de az elemző függvény alkalmazhatóságához mégis kénytelenek vagyunk egyetlen hálózat topológiát definiálni. Ezt az ellentmondásos helyzetet a fejezetben ismertetett elgondolás alapján sikerült feloldanom.

A probléma pontosításához vezessük be az alábbi jelöléseket.

1. Definíció. *Tekintsük pontok N, és élek L halmazát a következőképpen:*

$$N=\{p_i \mid i=1..m\} \text{ és } N\neq\emptyset$$

$$L=\{l \mid l=(p_j, p_k) \text{ ahol } j,k \leq m\} \text{ és } L\neq\emptyset$$

Az így kialakított objektumok halmazát H(N,P)–vel jelöljük és a továbbiakban ***hálózatnak*** *nevezzük.*

2. Definíció. *Tekintsük a $H_1(N_1,P_1)$ és a $H_2(N_2,P_2)$ hálózatot, melyekre teljesül $N_1=\{p_i \mid i=1..m_1\}$ $N_2=\{q_i \mid i=1..m_2, m_1 \geq m_2\}$ $N_1 \cap N_2=\emptyset$ $L_1 \cap L_2=\emptyset$*

A H(N,L) hálózatot a ***H_1 hálózat H_2-vel való bővítésének*** *nevezzük és* ***$H=H_1+H_2$*** *módon jelöljük, ha*

$$N=N_1 \cup N_2$$

$$L=L_1 \cup L_2 \cup L_3,\ L_3 \neq \varnothing$$

$$L_3=\{l \mid l=(p_j, q_k)\ ahol\ p_j \in H_1,\ q_k \in H_2,\ j=1..m,\ m \leq m_2\}$$

összefüggések érvényesek.

Jelölés: Egy H(N,P) hálózat objektumain definiált topológiát jelöljük a továbbiakban T(H)-val.

3. Definíció. *A T(H) topológiát helyesnek nevezzük, ha topológiai adatai konzisztensek, azaz nem tartalmaznak érvénytelen hivatkozásokat.*

Az adatok konzisztenciáját megszüntetheti például egy olyan csomópont törlése, mely még részt vett él objektum meghatározásában. Ebben az esetben a csomópont nélkül maradt él objektumok törlésével, vagy a hiányzó csomópont pótlásával az adatok konzisztenciája helyreállítható.

4. Definíció. *A T(H), $H=H_1+H_2$ topológia* ***T(H)-H_2****–vel jelzett* ***résztopológiáján*** *azt a topológiát értjük, mely a T(H) topológia H_2-beli éleinek eltávolításával és a keletkezett izolált csomópontok megszüntetésével áll elő.*

1. Állítás. *A T(H), $H=H_1+H_2$ topológia T(H)-H_2 résztopológiája helyes topológia.*

Bizonyítás:

Élek törlése nem befolyásolja a topológia helyességét, mert a szomszédsági információk a kezdő és végpontot definiáló csomópontok azonosítójával kerülnek tárolásra.

A csomópontok esetében a definíció szerint csak azok szűnnek meg, melyekre már nem illeszkedik él, tehát nincs már az adatok közt olyan él objektum, mely az eltávolított csomópontokra hivatkozna.

Következmény: *A T(H)-H_2 résztopológia alkalmas elemző műveletek végrehajtására.*

A 2. Definíció általánosításával egy hálózat többszörösen való bővítése határozható meg. A többszörösen bővített hálózaton definiált topológiából számtalan résztopológia keletkezhet a 4. Definícióban meghatározott módon. Az ilyenformán létrejött résztopológiákra teljesül az 1.Állítás alábbi, általánosított változata:

2. Állítás. *A T(H), $H=H_1+H_2+\ldots+H_n$ topológia T(H)-H_i résztopológiája helyes topológia minden i=1..n értékre.*

Bizonyítás:

A bizonyítás menetét nem befolyásolja, hogy a H hálózat többszörös bővítéssel keletkezett. Az állítás bizonyítása megegyezik az előző bizonyítás gondolatmenetével.

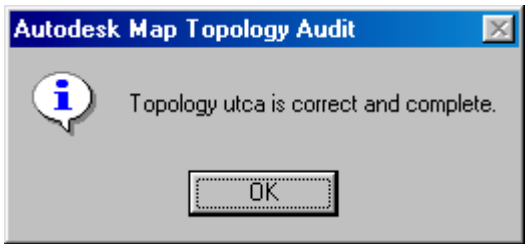
4.5.2. Elemzés megvalósítása résztopológián

Az előző fejezetben ismertetett elmélet gyakorlatban történő megvalósítása képezi a jelen fejezet tárgyát. Bemutatom azt a megoldást, mellyel biztosítható, hogy a projektben mindig az elemzések végrehajtásához szükséges résztopológia legyen jelen.

A probléma megoldásához a következő eszközök alkalmazási lehetőségeit használtam fel:

- Objektumok fóliák szerinti csoportosíthatósága.
- Forrásrajz objektumainak projektrajzba történő lekérdezése.
- Topológia projektrajzból való betölthetősége.
- Topológia állapotának (helyességének) lekérdezhetősége.
- Adott fólia objektumait törlő függvény definiálása (lásd 6.3. fejezet).
- Objektumadat tábla törölhetősége

A megvalósítás alapötletét az szolgáltatta, hogy ha egy topológiát alkotó objektumok mind a forrásrajzban mind a projektrajzban jelen vannak, akkor a memóriába való betöltésénél kiválasztható, hogy honnan kerüljenek az adatok a memóriába.



22. Ábra Információ a topológia állapotáról

A forrásrajz értelemszerűen tartalmazza a topológiát felépítő összes él és csomópont objektumot, valamint a topológia adatait. Autodesk Map felületen történő minden egyes betöltés alkalmával információt kapunk a topológia aktuális állapotáról is (22. ábra), de ez az információ bármikor lekérdezhető akár programkódból is. Forrásrajzból történő betöltés esetén az objektumok köre nem változtatható meg. Ha viszont meg tudjuk oldani, hogy a hálózatot alkotó objektumoknak csak egy olyan részhalmaza kerüljön bemásolásra a projektrajzba, melyből betöltött topológia állapotának lekérdezése során teljesnek és helyesnek bizonyul, úgy a projektrajz objektumain hálózat elemzés végezhető.

Hogyan válasszuk ki a forrásrajz objektumai közül a helyes résztopológiát alkotó objektumokat?

A forrásrajz erre a célra kialakított fólia szerkezete lehetővé teszi, hogy Tulajdonság lekérdezés definiálásával mindig a fenti igényeknek megfelelő objektumok kerüljenek a projektrajzba.

	Utca_hálózat	Hoz	TK_busz	TK_villamos	TK_troli	At
Semmi	✓	✓	×	×	×	×
Busz	✓	✓	✓	×	×	✓
Villamos	✓	✓	×	✓	×	✓
Troli	✓	✓	×	×	✓	✓
Busz+V	✓	✓	✓	✓	×	✓
Busz+T	✓	✓	✓	×	✓	✓
Vill.+T	✓	✓	×	✓	✓	✓
B+V+T	✓	✓	✓	✓	✓	✓

6. Táblázat

A 6. táblázat azt mutatja, hogy a felhasználhatónak megjelölt tömegközlekedési eszközöktől függően, hogyan változik a szükséges objektumokat tároló fóliák halmaza. A beállításoknak megfelelő fólián található objektumokat a *geometria_betolt* függvénnyel lehet a projektrajzba másolni.

A függvény szintaktikája:

(geometria_betolt *folia_lista*)

A paraméterek jelentése:

folia_lista A betöltendő fóliák elnevezéséből kialakított lista.

Visszatérési értéke: True

Ha a résztopológiát alkotó összes objektum jelen van a projektben, akkor a *tpm_acload* VLISP függvény megfelelő paraméterezésével a projektből tölthetjük be a topológiát, mely helyesnek fog bizonyulni, ezért elemzésnek alávethető.

Pszeudó kód segítségével összefoglalom a résztopológia kialakításának menetét (programkód az *utkere*s függvényben):

Előző műveletek végeredményeit tartalmazó fóliák ürítése.
Előző útvonalkeresést leíró tábla és topológia törlése.
Projektbeli Utca topológia helyességének vizsgálata.
Utca topológia memóriából való törlése.
Ha helyes volt a topológia
    ha engedélyezett a tömegközlekedés használata
        listát készít a betöltendő fóliákról, és
        üríti azokat, amelyikre nem lesz szükség.
    ha nem engedélyezett a tömegközlekedés használata
        üríti a tömegközlekedéssel kapcsolatos fóliákat
Ha nem volt helyes a topológia
    minden fóliát ürít,
    majd a beállításoknak megfelelőeket feltölti.

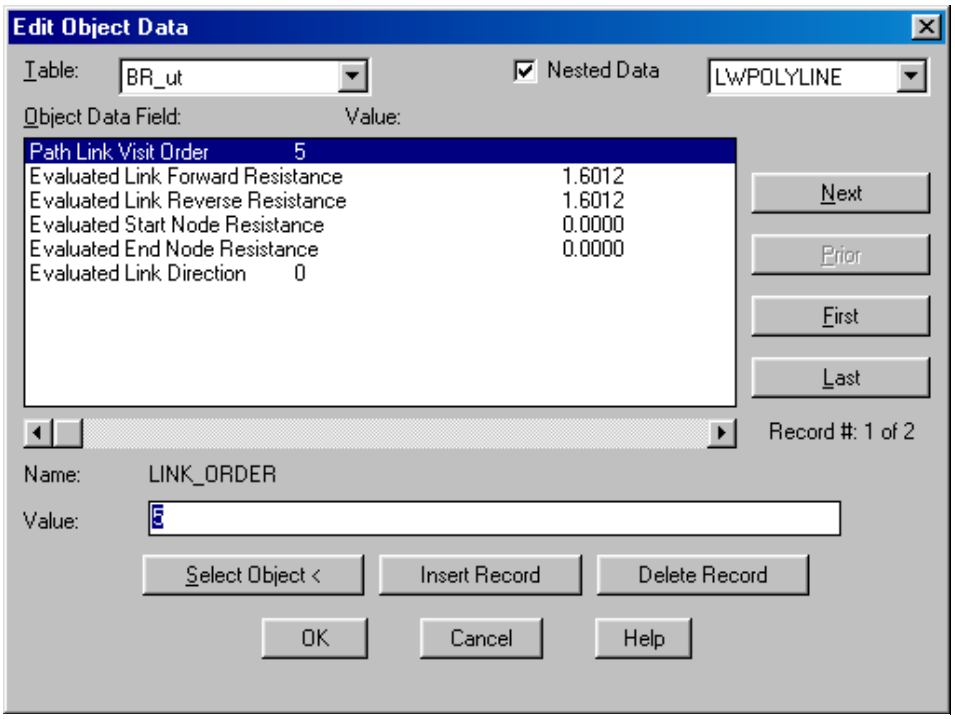
A legjobb útvonal meghatározásán jelen esetben a leggyorsabban megtehető útvonalat tekintettük, ezért a profile fájlban a következő kifejezés segítségével írjuk le az élek direkt és fordított ellenállásait.

(/ (/ .length 1000) :atlag_seb@sebesseg

A kifejezés felépítéséhez felhasználtuk a 4.2.2. fejezetben definiált *sebesseg* elnevezésű objektumadat táblát, mely rekordjai az egyes él objektumok bejárásához használható sebesség értéket tárolja.

Az elemzés végeredményeként egy *ut* nevű hálózat topológia jön létre a projektrajzon, az út bejárásához szükséges információk pedig egy BR_ut nevű objektumadat táblában (lásd 3.5. fejezet) kerülnek rögzítésre.

Minden él objektumhoz annyi rekord kerül csatolásra ahányszor részt vesz az útvonal felépítésében. A 23. ábrán egy olyan él objektumhoz tartozó rekord adatai láthatóak, melyen kétszer is végighaladunk az útvonal bejárása során. Erre utal a Next, First, Last nyomógombok aktív állapota, és az alattuk feltüntetett kifejezés, mely szerint két rekord közül jelenleg az elsőt látjuk.



23. Ábra Egyetlen él objektumhoz két rekord is tartozhat a BR_ut táblából

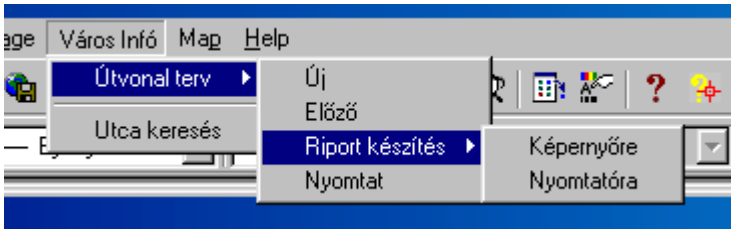
A profile állományban kérhető lenne az eredmény automatikus kiemelése, de jobbnak láttam egy saját függvény definiálását erre a célra. Az *utvonal_megjelenitese* függvény különböző színek és vonaltípusok használatával jeleníti meg az ajánlott útvonalat az Utvonal fólián. Amennyiben engedélyezett volt a tömegközlekedési eszközök használata, úgy a felhasználásra került jelölésmódokról egy jelmagyarázat is készül, mely a rajzterületen az útvonaltól balra, lent látható. Az útvonal során

érintett épület objektumok rajzai, és az aktuális nyelvű elnevezéseik is feltüntetésre kerülnek, majd utolsó lépésként úgy állítom be a rajz nézetét (lásd 6.2. fejezet), hogy minden információt hordozó objektum látható legyen.

4.5.3. Térkép és riport készítése

A célkitűzésben meghatározott felhasználás szerint, az alkalmazás egy fix helyen működő számítógépen érhető el, az előállított útvonal tervekre viszont nem csak helyben lehet szüksége a felhasználónak. Ez azt jelenti, hogy a hatékonyság növeléséhez biztosítanunk kell valamilyen módon, hogy az alkalmazás által előállított információ szabadon „elvihető" legyen.

Ennek megvalósításához a *Város Infó* nevű menüben elhelyezve két további almenüt építettem be, melyekkel az elkészült útvonal ajánlatról térképrészlet nyomtatása, illetve szöveges riport nyomtatása kezdeményezhető. A szöveges riport megjeleníthető a képernyőn is egy párbeszédpanel segítségével, illetve kérhetjük annak kinyomtatását is vagy közvetlenül a menüből, vagy a párbeszédpanelben elhelyezett nyomógomb segítségével.

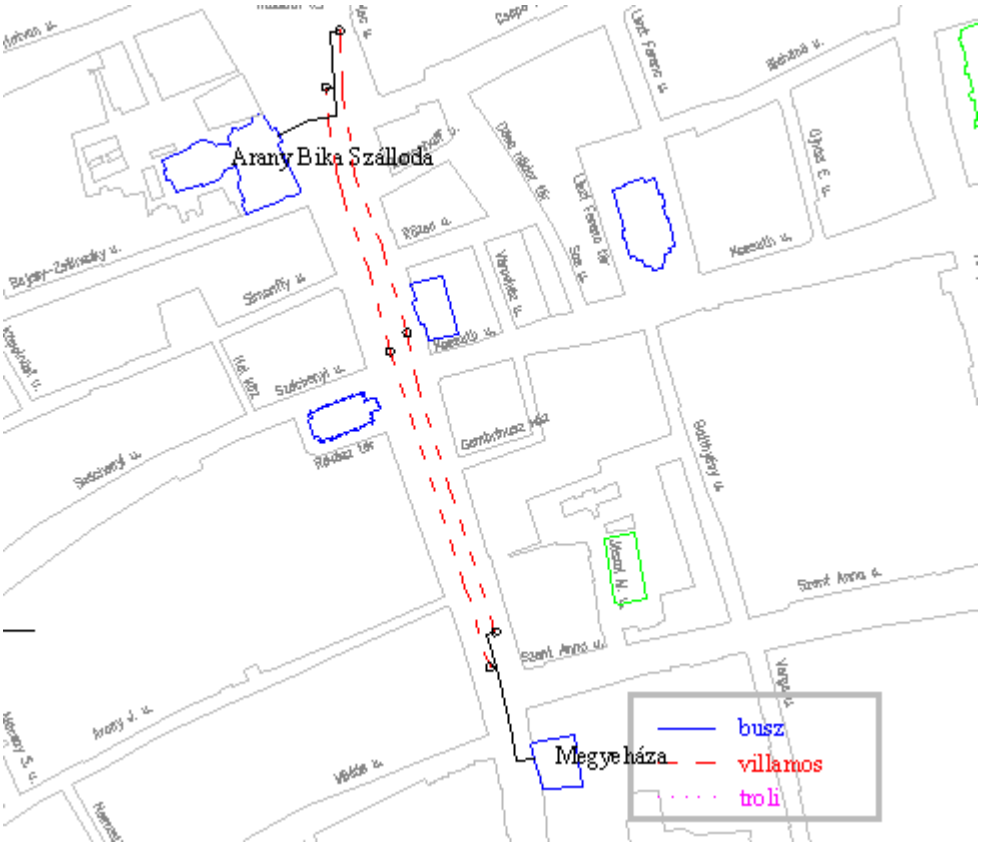


24. Ábra Az alkalmazás menürendszere

A Nyomtat menüpont segítségével egy olyan térképrészletet készíthetünk, ahol az utcákat körvonalaik képviselik, az útvonal megjelenítéshez alkalmazott vonaltípusok pedig a nem színes nyomtatón készült térképek esetén is megkönnyítik a felhasznált tömegközlekedési eszközök típusainak azonosítását.

Az útvonal szöveges leírásának elkészítését a *riport* elnevezésű függvény valósítja meg. A bejárt élek sorrendje a BR_ut objektumadat

táblából (23. ábra) kiolvasható ugyan, de a függvény megvalósítása során számos problémával meg kellett küzdenem.



25. Ábra A Megyeháza meglátogatásához villamost használatát ajánlja

Ízelítőül néhány:

- A tábla rekordjai kizárólag az objektumokon keresztül érhetőek el.
- Egy objektumhoz több rekord is tartozhat a táblából, hiszen például a kiindulási ponthoz visszafele vezető úton, újra érinthetünk egy korábbi élet.

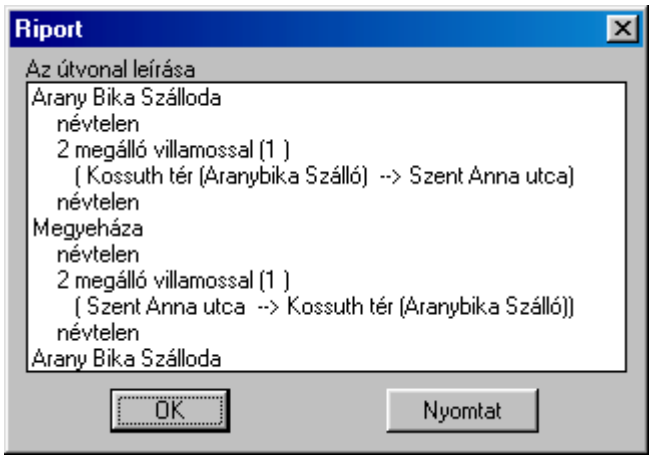
- A tábla nem tartalmaz információt arról, hogy mikor érkezem meg egy meglátogatandó célponthoz.
- Meg kell említeni az összes járatszámot, ami az adott útvonalrészen közlekedik.

Az útvonal éleiből képzett kiválasztási halmaz, és a BR_ut objektumadat tábla segítségével felépítettem egy speciális szerkezetű listát, melynek minden egyes eleme az útvonal egy-egy éléről tartalmazza az alábbi információkat:

(BR_ut_beli_sorrend – kiválasztási_halmazbeli_index – rekord_sorszáma)

Az így előállított lista (*utca_lista* névvel szerepel a forráskódban) rendelkezik az alábbi előnyös tulajdonságokkal:

- A *vl-sort* VLISP függvény segítségével az első tag szerint növekvő sorrendbe rendezhető. Ezzel az útvonal bejárását nyomon tudjuk követni.
- Közvetlen elérést biztosít az él objektumokhoz, és a hozzájuk csatolt objektumadatokhoz illetve külső adatbázis rekordokhoz egyaránt.



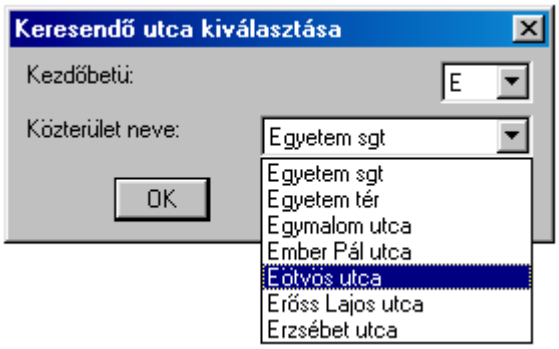
26. Ábra A 25. ábrán látható útvonalról készült riport

A riport elkészítésének a továbbiakban, az élek rendezett lista szerinti feldolgozása képezi az alapját. A megvalósított fóliaszerkezet nagyban elősegíti, hogy a végeredményt képező *utcanevek* elnevezésű lista előállítható legyen. A 26. ábrán a 25. ábrán bemutatott útvonalról készült riport olvasható, ahol a zárójelben feltüntetett számok a tömegközlekedési eszköz járatszámát jelzik.

4.6. Az alkalmazás egy további funkciója

A városon belüli eligazodáshoz gyakran szükségünk van egy-egy utca térképen való pontos elhelyezkedésének megtalálására. Egy digitális térképet tartalmazó alkalmazás fejlesztésekor ennek a funkciónak a megvalósítása kihagyhatatlan volt számomra, így ez is elérhető a legördülő menü második pontjaként.

Tekintettel az utcák igen nagy számára, a párbeszédpanelben előbb kezdőbetűt kell választani. A legördülő lista egyik elemére történő kattintást követően a közterület neveket tartalmazó lista máris feltöltődik az adott betűvel kezdődő utcanevekkel (27. ábra), melyekből egy kiválasztását engedélyezi az alkalmazás.



27. Ábra Keresendő utca kiválasztása

Az utcaneveket kezdőbetűnként egy-egy szöveges állomány tárolja, mely a külső adatbázisként használt Excel táblázatból keletkezett. A fájlok elnevezése minden esetben a tárolt utcák kezdőbetűjével egyezik meg. A kezdőbetűk felsorolásához így kézenfekvő volt az Utca mappában található állománynevek felhasználása.

A keresést SQL típusú, fólia tulajdonságot megváltoztató lekérdezés segítségével oldottam meg. Ennek köszönhetően a végeredmény külön fólián jelenik meg (Lek_utca), ezért alkalmazhattam a zoom parancs 6.2. fejezetben ismertetett formáját, ezzel biztosítva a lekérdezésnek eleget tévő objektumok rajzterületen való láthatóságának biztosítását. Egy újabb utca keresés végrehajtását követően az előző keresés eredménye törlődik a projektrajzról.

4.7. A többnyelvű alkalmazás

Napjainkban a többnyelvűség már alapvető követelmény a különféle alkalmazásokkal szemben. A többnyelvűség kifejezés mögött két külön fogalom húzódik meg:

1. Egy alkalmazásnak több változata is létezik a kommunikációhoz használt nyelv alapján.
2. Az alkalmazás használata közben lehetőség van a kommunikáció nyelvének megváltoztatására.

Úgy gondoltam, hogy az általam kifejlesztett alkalmazástól elvárható, hogy a másodikként megfogalmazott többnyelvűségnek is eleget tegyen. Sőt mindezt olyan formában valósítottam meg, hogy a 4.7.2. fejezetben leírt útmutatást követve a használható nyelvek köre tetszőlegesen tovább bővíthető.

4.7.1. Megvalósítás

A felhasználó a menüsorban találkozik először az alkalmazással. A menükről általánosságban a 3.6.1. fejezetben már írtam. Egyetlen menüfájl több, tetszőleges típusú menü definíciót is tartalmazhat, így az alkalmazásomhoz szükséges menük forráskódját egyetlen állomány (*smenu.mns*) tartalmazza. A különböző menü típusokat azonban különböző szekciókban kell elhelyezni, melyek akár további almenüket is tartalmazhatnak.

A többnyelvűség megvalósításához a legördülő menüt és annak súgóját is minden, használni kívánt nyelven el kell készíteni úgy, hogy a legördülő menük egyedi azonosítóval rendelkezzenek. A kommunikáció nyelvének megváltoztatásához az ikonokból álló eszköztár létrehozása tűnt

célszerűnek. Egyrészt az ikonokon feltüntetett zászlók szavak nélkül utalnak funkciójukra, másrészt számos alkalmazás használta már ezt, vagy ehhez hasonló megoldást a nyelv megváltoztatására.

Az eszköztár egy ikonjára történő kattintás a következő eseményeket vonja maga után (amennyiben az éppen aktuálistól eltérő nyelvet választottunk):

- Tizenkettedik legördülő menüként betöltésre kerül az eszköztár segítségével kiválasztásra került nyelvű menü. Ezzel gyakorlatilag az előző nyelvű menü helyére újat töltöttünk eltüntetve ezzel az előzőt. (A tizenkettedik hely megválasztása önkényes volt, a beállítás az aktuális környezet függvényében tetszőlegesen megváltoztatható, a 4.8. fejezetben ismertetett módon.)
- Az aktuális nyelvre utaló kettő hosszú karakterláncot egy rendszerváltozó (*users1*) tárolja. Más nyelvre való áttéréskor a rendszerváltozó értékét is aktualizálni kell.
- A *nyelvet_cserel* függvény meghívásával a memóriába töltődnek az aktuális nyelvű hibaüzenetek, és a riport készítéséhez szükséges szavak. Egyúttal a rajzterületen már meglévő feliratok is megújulnak.

A párbeszédpanelek felépítését tartalmazó *dcl* kiterjesztésű állományokból is szükség van nyelvenként egy-egy változatra. A párbeszédpanelek indítása előtt a *users1* rendszerváltozó tartalmának lekérdezésével előállítható a betöltendő dcl állomány neve. Például: mitnezmeg_hu.dcl.

Ugyanezzel a technikával kerül megállapításra a *hiba_uzenetek* illetve a *szokincs* változó aktualizálásához megnyitandó állományok elnevezése is. Mindkét változó lista típusú, így az alkalmazás a lista elemekre – tartalmuktól függetlenül – azok indexének felhasználásával tud hivatkozni.

A párbeszédpanel működéséhez használt *latni_lb* változó felépítését, illetve annak a megfelelő szöveges állományból való előállítását a 4.4. fejezetben már ismertettem.

Jelenleg az alkalmazáshoz szükséges menü, és maga az alkalmazás is magyar nyelven indul. Ez természetesen megváltoztatható, a következő két lépés segítségével:

1. Az smenu.mnl állományban a (menucmd (strcat m_hely "=TOUR.POP2")) függvényt tartalmazó sorban a POP2 magyar nyelvű menüt azonosító címke helyett a megfelelőt kell megadni. (Az angol nyelvű például a POP3.)
2. Ugyanebben az állományban a következő sorban található (setvar "users1" "HU") függvény utolsó paramétereként használt, a nyelvre utaló két karaktert is ki kell cserélni.

A változtatások érvénybe lépéséhez mentsük el az állomány tartalmát, majd indítsuk újra az Autodesk Map-et.

4.7.2. Az alkalmazás új nyelvre való megtanítása

Az új nyelv elsajátítása gyakorlatilag címkék, szavak, üzenetek, leírások megfelelőinek a megadását jelenti az alkalmazás számára. A könnyebb érthetőség kedvéért, a továbbiakban feltételezem, hogy a német nyelvvel szeretnénk bővíteni a kommunikáció lehetséges nyelveit. A nyelvet azonosító karaktersorozatnak a DE-t választom, alkalmazkodva ezzel az Interneten használt jelölésekhez.

Új nyelvvel való bővítés előtt mindig készítsünk biztonsági másolatot a szerkesztendő állományokról. Az ebben a fejezetben leírt műveleteket pontosan, precízen kell végrehajtani ahhoz, hogy az alkalmazás működésében ne történjen nem kívánatos változás. Helytelen működés esetén használjuk a biztonsági másolatokat.

A menü bővítéséhez szükségünk lesz az új nyelvet szimbolizáló zászlóról két bitképre, melyek felbontása 16x16, illetve 32x32 legyen [1]. Elnevezésükben feltétlen jelenjen meg a nyelvre utaló két karakter, illetve a felbontást jelző szám. (Például flag_de16.bmp, flag_de32.bmp.) Az eszköztárakhoz szükséges bitképeket az Autodesk Map a Support mappában keresi, ezért érdemes oda elhelyezni őket, ellenkező esetben külön gondoskodnunk kell róla, hogy megtalálja az állományokat a rendszer. (Tools menü Options parancs File oldal)

Következő lépésként a menü felépítését definiáló smenu.mns állományt nyissuk meg egy tetszőleges szövegszerkesztővel, s végezzük el az alábbi változtatásokat:

- Az utolsó legördülő menü definícióját követően (***TOOLBARS címke elé) szúrjuk be egy korábbi menü

definícióját, majd javítsuk ki a címkéjét ***POP4-re. (A számjegy értelemszerűen mindig eggyel növekszik az újabb nyelvek megadásakor.)

- A menüpontokat definiáló részben minden sorban fordítsuk le a kívánt nyelvre a szögletes zárójelben található kifejezéseket. A zárójelben esetenként előforduló vezérlő karaktereken nem szabad változtatni.
- A sorok első, azonosításra használt címkéjén elegendő a nyelvre utaló utolsó két karakter kicserélése.

ID_Terv_Nyomtat_EN [<-Print](terkepet_nyomtat)
ID_Terv_Nyomtat_***DE*** [<-***Drücken***](terkepet_nyomtat)

- Az eszköztárat is bővítsük egy újabb elemmel, melyre kattintva áttérhetünk német nyelvre. Ehhez megint lemásolhatunk egy előző definíciót, majd a dőlt, vastag betűs részeket átírjuk.

ID_Nyelv_***DE*** [_Button("***Deutsch***";"flag_***de***16.bmp","flag_***de***.bmp")]
^C^C(if (/= (getvar "users1") "***DE***") (progn (menucmd "P12=TOUR.POP***4***")(setvar "users1" "***DE***")(nyelvet_cserel)))

- Végül az újonnan keletkezett (a sorok elején található) címkékhez tartozó, státuszsorban alkalmanként felbukkanó súgó szövege adható meg, az azonosítót követően szögletes zárójelben.

A párbeszédpanelek felépítését tartalmazó dcl állományokról készítsünk egy másolatot. Cseréljük ki az állomány nevében található utolsó két karaktert értelemszerűen az új nyelvre utaló jelzésre, majd nyissuk meg egy tetszőleges szövegszerkesztővel. A párbeszédpanelen megjelenő szöveges feliratok minden mozaik esetében a

label = felirat

szintaktikával adhatóak meg. Így csak az egyenlőségjel jobb oldalán található kifejezéseket kell átfordítanunk a kívánt nyelvre, majd gondoskodnunk a változtatások elmentéséről.

Az egyes látnivalókról a párbeszédpanelben megjelenített rövid leírások nyelvenként külön állományban tárolódnak, viszont azonos

mappában. A txt kiterjesztésű állományok elnevezése kötött, az alábbi szabályszerűség szerint:

látnivaló-ötbetűs-azonosítója+aláhúzásjel+kétbetűs-nyelvre-utaló-jel

Tartalmát illetően szigorú előírás, hogy csak két sorból állhat:

- A látnivaló elnevezése.
- Rövid leírása egyetlen (tetszőleges hosszú) sorban.

Az alkalmazás további három szöveges állományt használ még a kommunikációhoz szükséges különböző szavak tárolására. Ezek közül kettő (hiba, szavak) tartalmát, a szavak sorrendjének változtatása nélkül egyszerűen le kell fordítani a szükséges nyelvre, majd a nyelvre utaló jelzéssel elmenteni az állományokat.

A latniv_de.txt elkészítéséhez minden sorban csak az utolsó tagot, azaz a látnivaló elnevezését kell lefordítani. A sorok sorrendje megváltoztatható. Érdemes azonban az elnevezések szerint betűrendbe állítani, mert akkor a párbeszédpanelben is betűrendben fognak következni egymás után a listadoboz elemei.

A változtatások kipróbálásához távolítsuk el az smenu.mnc és smenu.mnr állományokat, és így indítsuk el az Autodesk Map-et. A szoftver a módosításainkat tartalmazó smenu.mns alapján induláskor elkészíti az új mnc és mnr állományokat. Ha az eszköztárat sikeresen bővítettük, akkor az újabb kis zászló már automatikusan megjelenik a többi mellett.

Amennyiben nincs szükségünk az alkalmazás által ismert valamelyik nyelvre, úgy az előző két fejezetben ismertetett változtatások adott nyelvre vonatkozó részeit szüntessük meg. Ezzel a lépéssel háttértárat szabadíthatunk fel számítógépünkön, s csökkentjük az eszköztár ikonkészletét.

4.8. Az alkalmazás telepítése

Az alkalmazás használatbavétele körüli teendőket kezdjük azzal, hogy a CD melléklet DB mappáját másoljuk fel a számítógépünkre tetszőleges helyre változatlan elnevezéssel. A Support mappa állományai értelemszerűen az Autodesk Map telepítési mappájából nyíló, ugyanilyen

elnevezésű mappájába helyezendők el, míg a Map mappa egyetlen állományának az Autodesk Map telepítési mappájába kell kerülnie.

Ezt követően Autodesk Map felületen a következő lépéseket kell végrehajtani:

- Hozzuk létre a PHD elnevezésű meghajtó álnevet, mely a számítógépünkre felmásolt DB mappára mutat, vagy módosítsuk megfelelően az smenu.mnl állományban található, hiányzó álnév pótlására használt *ade_aliasadd* függvény harmadik paraméterét. (Az állományban megjegyzés sor ad útmutatást a változtatás elvégzéséhez.)
- Az Tools/Options menü Files lapján bővítsük a Support file Search Path listát egy új elemmel, mely az alkalmazás mappájára mutat.
- A Tools/Customize/Menus menüpont segítségével töltsük be a DB mappa smenu állományából a TOUR menücsoportot, majd 12. menüként szúrjuk be a magyar nyelvű Város Info-t.

A beállítások elvégzését követően indítsuk újra az Autodesk Map-et, így a saját menünk betöltésekor a hozzá tartozó lsp kiterjesztésű fájlban tárolt függvények is végrehajtódnak.

Mielőtt használatba vennénk az alkalmazást, el kell készíttetni a programmal a külső adatbázishoz való kapcslódást leíró udl állományt. Ehhez nyissuk meg a DB_varos.dwg forrásrajzot, majd az alkalmazás telepítési mappájában található db_utca.xls állományt vontassuk a munkatér intéző területére. Az eredményes csatolás és csatlakozás eredményeképpen a csatolási sablonról eltűnik a piros kereszt jel. A rajzfájl ezen állapot mentsük el ezt követően.

Az elemző műveletet vezérlő profile állomány elérését nem sikerült szabadon szabályozni az alkalmazásból, ezért még az első elemzés végrehajtása előtt tegyük meg a következő lépéseket a forrásrajzon:

- Töltsük be a memóriába az *Utca* nevű topológiát.
- Válasszuk a Map/Topology/Network Analysis menüpontot.
- A megjelenő párbeszédpanelben kattintsunk a Load gombra, majd keressük meg az Autodesk Map telepítési mappájába másolt utv_p.tpf állományt, s nyissuk meg az Open gombbal.

- Az adatok betöltését követően Cancel gomb segítségével hagyjuk el a párbeszédpanelt.

E lépések eredményeként az alkalmazás által készített, s az Autodesk Map telepítési mappájában tárolt profile állomány elérhető lesz a MAPANTOPONET parancs számára. Sikertelen útvonal meghatározás esetén e lépéseket kell megismételni.

Az első térképrészlet nyomtatásakor el kell végeznünk a szükséges beállításokat a sikeres eredmény eléréséhez. Ezt az Autodesk Map elraktározza, s a továbbiakban nem kell külön figyelmet fordítani erre.

Végül a VInf_p.dwg projektfájl megnyitását követően bátran kipróbálhatjuk a menürendszer segítségével az alkalmazás működését. A projekt bezárásakor mindig rákérdez a változtatások elmentésére, ami jelen esetben nem indokolt, ezért válasszuk a Nem gombot.

Az alkalmazást nem muszáj természetesen 12. menüként használni. Ennek megváltoztatásához csupán az smenu.mnl állományban definiált *m_hely* nevű változó tartalmát kell a jelenlegi p12-ről a szükségesre átírni, majd az új helyre feltölteni egyszer a menüt.

A fejezetben ismertetett lépések közül lehetne ugyan néhányat automatizálni, de tekintettel a dolgozat az alkalmazás minden részletét feltáró jellegére indokoltnak láttam ettől eltekinteni.

„…ne becsüljük le diadalainkat.
Nem mindig következik be a legrosszabb…”
Lord Charles Percy Snow
angol természettudós,
regény- és esszéíró

5. Pont a poligonban algoritmus

A *pont a poligonban teszt* egy olyan alapvető fontosságú algoritmusra épül, mely gyakran kerül felhasználásra mind önállóan, mind pedig összetett, bonyolultabb algoritmusokban (például a poligon fedvényezési műveletben). Számos térinformatikai alapismereteket tartalmazó könyvben is megtalálhatjuk az ismertetését, de sajnos gyakorta elnagyolva, és nem minden esetben jól működő változata lelhető fel. Az alábbi algoritmus alapötlete azonos az említettekkel, de egyéni megoldásokat is tartalmaz, mellyel egyúttal a felhasználhatósági körét is sikerült bővíteni.

Az alapprobléma:	Adott egy poligon és egy pont, melyről el kell döntenünk, hogy a poligonon belül vagy kívül található.
A válasz meghatározását elősegítő alapötlet:	Számoljuk meg, hogy az adott pontból kiinduló, pozitív irányba mutató függőleges félegyenes hányszor metszi a poligon oldalait. Amennyiben ez a szám páros, akkor a kérdéses pont külsőnek bizonyult, amennyiben páratlan, akkor pedig belsőnek. (A pontból kiinduló függőleges félegyenes irányának megválasztása a végeredményt nem befolyásolja.)

Az alapötlet felhasználásával elegendő tehát egy ciklust szervezni, mely a poligon oldalai és a félegyenes metszéspontjainak számát határozza meg. (Ez gyakorlatilag két egyenes metszéspontjának meghatározásával történik,

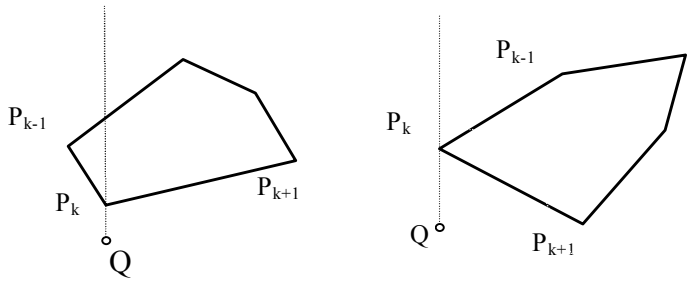
de annak egy igen speciális esete, tekintettel az egyik egyenes függőleges voltára.)

A tényleges metszéspont számítások száma az alábbi összefüggés alkalmazásával jelentősen csökkenthetőek.

Az f(x)=u egyenes akkor és csak akkor metszi a $P(x_1,y_1)$ és $Q(x_2,y_2)$ pontok által meghatározott szakaszt, ha

$$(x_1-u)*(u-x_2) \geq 0.$$

Az alábbi összefüggést feltételként alkalmazva, még a ciklus elején ki lehet szűrni azokat az érdektelen oldalakat, melyek teljes egészében a félegyenes azonos oldalán fekszenek, ezért biztosan nincs közös pontjuk a félegyenessel. A félegyenessel párhuzamos, tehát függőleges oldallal sem kell törődnünk, mert vagy a félegyenes azonos oldalán található, vagy egybeesik a félegyenessel.

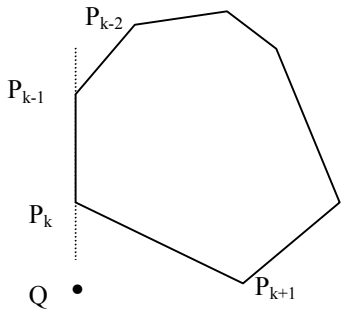


28. Abra Problémás esetek

Problémát okoznak viszont a metszéspontok számának meghatározásában 28. és a 29. ábrán bemutatott esetek, amelyeken a Q pontból indított félegyenesre illeszkedik egy vagy két poligon csúcs. Ezekben az esetekben azt kell eldöntenünk, hogy a csúcspontban való metszéspont növeli-e a metszéspontok számát, és ha igen, akkor milyen módon.

A megoldáshoz az a felismerés vezetett, hogy a metszések számának csak abban az esetben kell növekednie, ha – a 28. ábra jelöléseit használva – a P_{k-1} és a P_{k+1} koordinátájú pontok a félegyenes különböző oldalain helyezkednek el. Ez jelen esetben két szomszédos oldal, közös csúcstól

különböző csúcspontjainak vizsgálatával dönthető el. A 29. ábrán látható esetben viszont már a P_{k-2} és a P_{k+1} pontok félegyeneshez való viszonyát kell vizsgálnunk, azaz nem szomszédos oldalakhoz tartozó csúcspontok helyzete lesz döntő fontosságú.



29. Ábra

Általánosságban a következő útmutatás adható az ilyen helyzetek megoldására:

- Az első (és a továbbiakban minden páratlan számú) alkalommal, amikor a vizsgált oldal egyik csúcspontja illeszkedik a félegyenesre nem növeljük a metszések számát, csak elraktározzuk egy változóban, hogy az oldal a félegyenes melyik oldalán található.
- A második (és a továbbiakban minden páros számú) ilyen alkalommal csak akkor növekszik a metszések száma, ha a vizsgálatban résztvevő poligon oldal a változóban tárolt értékhez képest a félegyenes másik oldalán található.

A legjobb megoldásnak egy külön függvény definiálása tűnt, mely a problémás esetekben az alábbi pszeudo kód alapján eldönti, hogy folytatódjék a számítás, vagy jöhet a következő oldal vizsgálata.

ha az oldal valamelyik csúcsa a félegyenesre illeszkedik
ha volt már ilyen eset
és ugyanaz az oldal ->NEM kell számolni
és másik oldal -> IGEN, kell számolni
ha nem volt még ilyen eset
helyzet változóba rögzítése, és NEM kell számolni

Ha az előbbi függvény igen választ ad vissza, akkor a kiszámolt tényleges metszéspont y koordinátájának még a vizsgált pont y koordinátájától nagyobbnak kell lennie ahhoz, hogy valóban növelje a metszéspontok számát.

A teljes algoritmust a következő pszeudó kód írja le:

ha az oldal nem függőleges és
az oldal csúcspontjai a félegyenes különböző oldalára esnek és
a szamol függvény igen-t ad vissza
metszéspont y koordinátájának meghatározása
ha ez nagyobb mint a ponté
metszéspont száma nő

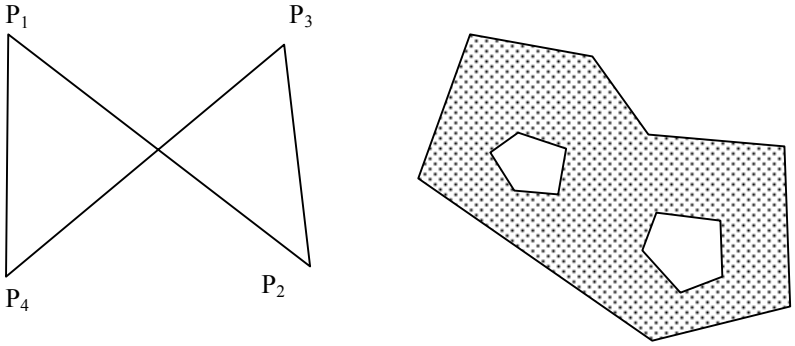
A CD mellékleten található VInf.lsp tartalmazza mind a *szamol* függvény, mind az ezt felhasználó pont_poligonban függvény teljes Visual LISP forráskódját, melyek szintaktikája:

(szamol *x xx u irany*) ahol

x, xx :	vizsgált poligon oldal csúcspontjainak első koordinátái
u:	vizsgált pont első koordinátája
irany:	előjelével jelzi, hogy melyik oldalon volt az előző alkalommal a problémás oldal

(pont_poligonban *p poligon*)

p:	pont objektum koordinátája
poligon:	poligon rajz objektum azonosítója



30. Ábra Különleges poligonok

Az így pontosított algoritmus a legáltalánosabb esetekben is jól működik, akár konkáv poligonokról, nem egyszerű poligonokról (nem csak a szomszédos oldalaknak van közös pontjuk), vagy a szigetet tartalmazó poligonokról legyen szó. Ezek a speciálisnak mondható poligonok a gyakorlatban is előfordulhatnak elég csak az övezetekre (pufferekre) vagy törpeállamot tartalmazó országok térképére, esetleg nyitott, belső udvart tartalmazó épületek (pl. Református Kollégium) alaprajzára gondolnunk.

„… az alkotás értelme nem a kor és a technika által meghatározott formális tényezőkben, hanem az ember alkotóerejének megnyilatkozásában rejlik.”
Moholy-Nagy László
(festő, művészetteoretikus)

6. Új AutoCAD parancsok

Még az AutoCAD fejlesztői is elismerik a dokumentációkban, hogy a parancsok definiálásakor ők sem gondolhatnak mindenki igényére, ezért bátran ajánlják a testreszabási eszközök használatát. Ennek egyik leggyakoribb formája, amikor új parancsok készítésével egészítjük ki a számunkra tátongó űrt.

Ebben a fejezetben néhány általam definiált parancsot ismertetek, melyekre az alkalmazás kifejlesztése közben gyakran szükségem volt. Jellegüket tekintve széles körben felhasználhatóak, ezért indokoltnak láttam mások számára is elérhetővé tenni. Ahol volt értelme ott elkészítettem a parancssorból használható változatot is, mely párbeszédpanel használatával megkönnyíti a felhasználó dolgát. A közölt Visual LISP szintaktikák elsősorban alkalmazásokból való használatra készültek, problémamentes futtatásuk nagyobb precizitást igényel.

A címben azért csak AutoCAD parancsot említek, mert térinformatikai vonatkozás nélkül is használhatóak a parancsok, ugyanakkor jól tudjuk, hogy ami igaz AutoCAD-re, az igaz Autodesk Map-re is, hiszen a Map gyakorlatilag az AutoCAD kiterjesztése.

6.1. Nagyítás kiválasztási halmaz elemeire

A kiválasztási halmaz egy vagy több kiválasztott objektum olyan halmaza, melyek egyszerre szolgálnak egy függvény bemeneteként. Azaz a függvény minden halmazbeli objektumon végrehajtódik, annak csupán egyetlen meghívásával.

Tekintsük át először is a kiválasztási halmazok leggyakrabban előforduló keletkezési módjait (a kizárólag Autodesk Map környezetben érvényeseket Map megjelöléssel különböztetem meg).

- Manuális kiválasztás
- Quick Select párbeszédpanel
- Lekérdezés, ha a Map/Options/Query lapon a Create selection set from queried objects jelölő négyzet be van jelölve. (Map)
- Visual LISP függvények (Map)
  - tpm_elemss
  - ade_editlocked
  - ade_editnew
  - ade_editlockederased
  - map_dwgbreakobj
  - map_dwgtrimobj
  - map_topoclose
  - map_topocomplete
- kiválasztási halmazt definiáló VLISP függvény: ssget

A manuális kiválasztástól eltekintve az összes többi esetben nincs semmiféle biztosíték arra, hogy a kiválasztási halmaz elemei a képernyő területén láthatóak. Magyaráznom nem szükséges, azt hiszem, hogy igény viszont van rá.
Ezért definiáltam a következő függvényt:

(zoom_kiv_hz *kiv_hz_az*)

Egy paramétere van:

kiv_hz_az ami a kiválasztási halmaz azonosítóját jelenti.

Visszatérési értéke:

nil

A megvalósítás során a kiválasztási halmaz minden elemének megállapításra került a minimális befoglaló téglalapja, melyek segítségével meghatározható a kiválasztási halmaz objektumainak minimális befoglaló téglalapja.

Az így kapott értékek felhasználásával kerül meghívásra a *zoom* parancs, mely eredményeképpen a kiválasztási halmazban található objektumok mindegyike a rajzterületen láthatóvá válik.

A túlzott ráközelítés egy korrekciós tényező beiktatásával van kivédve.

6.2. Nagyítás fólia objektumaira

Az AutoCAD *zoom* parancsa igen sokféle módon használható, de többünk hiányolta belőle a fóliák objektumaira való nagyítás lehetőségét. Más térinformatikai rendszerben már használtam hasonló szolgáltatást, így kellő tapasztalattal rendelkeztem a megvalósításához.

A következő helyzetekben tesz jó szolgálatot a fólia, vagy fóliák objektumaira kérhető nagyítás:

- A fólia objektumai a rajz terjedelméhez képest kis területre korlátozódnak
- Lekérdezés feltételeinek eleget tevő objektumokon fólia tulajdonság változtatást hajtottunk végre.
- Topológia elemzés végeredményét külön – eddig még üres – fólián jelenítjük meg.

A fólia objektumaira történő nagyítás megvalósításához jól használható a 6.1. fejezetben ismertetett függvény, mely kiválasztási halmazokkal dolgozik.

Kiválasztási halmazok alkalmazásokból történő létrehozására alkalmas a Visual LISP *ssget* függvénye [1, 4], melynek visszatérési értéke a kiválasztási halmaz azonosítója, vagy a halmaz sikertelen létrehozás esetén *nil*.

Számtalan szintaktikai formája közül számunkra most az alábbi a legfontosabb:

(ssget "_X" szűrő)

A formula gyakorlatilag a szűrőnek megfelelő összes objektumot a kiválasztási halmazba helyezi. A szűrőt alkotó feltételek asszociációs lista formájában adhatóak meg, s tartalmazhat logikai csoportosítást megvalósító operátorokat is.

A definiált függvényem VLISP szintaktikája:

(zoom_folia *folia_lista*)

Egy paramétere van:

folia_lista nagyítandó fóliák elnevezését tartalmazó lista

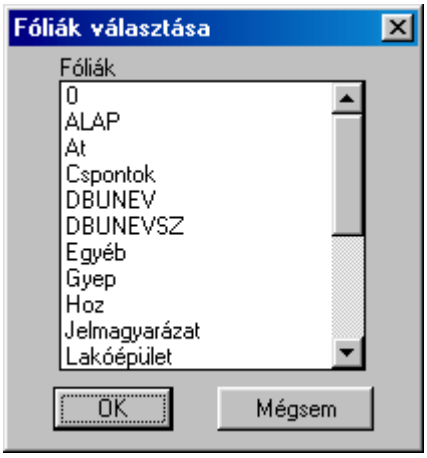
Visszatérési értéke:

T vagy *nil*

A függvény működése során a paraméterként kapott listából elkészíti az *ssget* függvény számára az alábbi felépítésű szűrőt:

(-4 . "<OR")(8 . *folia_nev1*)(8 . *folia_nev2*)…(-4 . "OR>")

A szűrő alkalmazásával így a függvény egy olyan kiválasztási halmazt állít elő, melynek objektumaira szeretnénk nagyítani. Ezt a problámát pedig már megoldottuk a 6.1. fejezetben ismertetett *zoom_kiv_hz* függvény definiálásával, tehát csak meg kell hívnunk azt a függvényt az *ssget* által visszaadott azonosító paraméterként való felhasználásával.



31. Ábra

A parancssori változat előnye, hogy a fólialistát a 31. ábrán látható párbeszédpanel segítségével határozzuk meg, elkerülve ezzel az esetleges elírásokból adódó problémákat.
A parancs formája:

zoom_f

A párbeszédpanel kezelését egy külön függvény, a *folia_valasztas_db* valósítja meg, mely visszatérési értékként a kiválasztott fólia nevekből előállított listát, vagy a *nil* értéket szolgáltatja. A függvény az aktuálisan megnyitott rajzállomány adataiból dolgozik, a lista elemei nem előre definiált értékek.

A párbeszédpanelben feltüntetett fóliák közül természetesen többet is kiválaszthatunk, hiszen a Visual LISP változat is fólialistával dolgozik. A kiválasztást követően csak a lista elemek indexe kérdezhető le, melyekkel még azonosítani kell a megfelelő fólia elnevezéseket.

6.3. Fólia összes objektumának törlése

Az alkalmazás fejlesztése közben gyakran szükségem volt arra, hogy eltávolítsak egy-egy fólián elhelyezkedő összes objektumot. AutoCAD környezetben ez több módon is megoldható, igaz, hogy több lépéssel. Alkalmazásból viszont célszerűnek tűnt egy függvényt definiálni erre a célra, mégpedig a lehető legáltalánosabb felhasználást is lehetővé téve.
A függvény Visual LISP szintaktikája:

(urit *folia_lista*)

Egy paramétere van:

folia_lista	Törlendő objektumok fóliáinak elnevezését tartalmazó lista.

Visszatérési értéke:

T	Ha minden objektum eltávolítása sikeres volt.
nil	Ha maradt objektum a listabeli fóliákon, valamilyen hibajelenség következményeként.

A függvény működése során a paraméterként kapott fólialista segítségével kiválasztási halmazt definiál, mely a listabeli fóliákon található összes objektumot tartalmazza. A kiválasztási halmaz keletkezése a 6.2. fejezetben ismertetett módon történik. Ezt követően egy ciklus segítségével eltávolítja a függvény a halmazhoz tartozó objektumokat.

A visszatérési érték meghatározásához mintegy ellenőrzésként újra kiválasztási halmazt definiálok a korábban meghatározott szűrő segítségével. Ezzel gyakorlatilag megállapításra kerül, hogy maradt-e objektum a megjelölt fóliákon, s ennek alapján kerül megállapításra a függvény visszatérési értéke is.

A parancsot elérhetővé tettem a parancssorból is a következő formában:

urit_f

A szükséges fólialista előállításához a 6.2. fejezetben ismertetett *folia_valasztas_db* függvény segítségével megvalósított párbeszédpanel (31. ábra) használható. E függvény visszatérési értéke változtatás nélkül alkalmas a parancs Visual LISP szintaktikájában szükséges paramétereként való felhasználásra.

„Az ember sohasem azt veszi észre
amit már elvégzett,
csak az lebeg a szeme előtt,
amit még tennie kell…”
Marie Curie
lengyel származású francia vegyész

7. Utószó

Ritkán készül minden szempontból befejezett alkotás. A dolgozat központi témáját alkotó városi tömegközlekedés modellezésének vizsgálatára és gyakorlati megvalósítására talán még inkább jellemző az előbbi megállapítás. A külső körülmények, elvárások folyamatos változása, a technika rohamos fejlődése állandóan új elvárásokat támaszt az alkalmazásokkal szemben. Ebben a fejezetben az „utolsó szó jogán” szeretném röviden felvázolni a további kutatási irányról való elképzeléseimet.

A 4. fejezet elején már említett városszéli nagy parkolók megépítése és üzemeltetése teljesen átalakítja egy város közlekedési rendszerét, így ennek megvalósulása esetén a dolgozatban felépített modell is átalakításra szorulna. Az elemző függvények helyes alkalmazása is újabb feltételeken alapulna, mely maga után vonná az összes megjelenítéssel, riporttal kapcsolatos függvények újragondolását is.

A CD mellékleten elhelyezett adatbázis nem tekinthető még teljesnek, célja egyelőre elsősorban a kifejlesztett függvények működésének a bemutatása volt. Ennek a célkitűzésnek viszont úgy gondolom, hogy maradék nélkül eleget tudott tenni, hiszen az alkalmazás szempontjából lényeges adatokból kellő mennyiséget tartalmaz.

További terveimben az alkalmazás Internetes változatának kifejlesztése is szerepel. Ehhez tudnunk kell, hogy az Autodesk cég térinformatikai szoftverei egymásra épülő teljes körű térinformatikai rendszert képeznek, melyben az Autodesk MapGuide-nak központi szerep jutott. A háromtagú szoftvercsalád kifejezetten a térinformatikai felhasználók adatmegosztási igényeit elégíti ki a meglévő digitális térképekből szabványos Web-böngészővel megtekinthető HTML dokumentumok generálásával és kezelésével.

8. Összefoglalás

A dolgozatban bemutatott eredmények egyetlen központi téma köré csoportosíthatóak, mely nem más, mint a városi tömegközlekedés modellezése és elemzése közben felmerült problémák megoldása Autodesk Map környezetben.

A dolgozat a kutatási környezet pontos meghatározásával kezdődik. A vektor alapú térinformációs rendszerek közül a topológiai adatmodell alkalmas a legváltozatosabb adatelemzések végrehajtására. Az Autodesk Map pontosan egy ilyen, topológiai modellen alapuló térinformatikai szoftvertermék, így a program eszközeinek ismertetésével a topológiai modell részletezése is megvalósult.

Az eredmények szempontjából kiemelt jelentőséggel bíró hálózat topológiát és annak elemző műveleteit kritikai elemzésnek vetettem alá.

Az eredmények gyakorlati bemutatásához igénybe vettem az Autodesk Map testreszabási lehetőségeit is, ezért előzetesen áttekintettem a legfontosabb tudnivalókat a saját menü, párbeszédpanel definiálásáról, illetve a Visual LISP fejlesztői környezetről.

Az inhomogén topológiák elemzésénél felmerülő igény kielégítésére kidolgoztam a hálózat résztopológia fogalmát. Ennek szemléltetésére tökéletesen alkalmas egy olyan városi tömegközlekedési modell, melyben definiálható hálózat topológián értelmezett elemző műveletek segítségével következő típusú kérdésekre keressük a választ:

- Melyik a például csak busz használatával leggyorsabban megtehető olyan út, mely egy adott kiindulási pontból előre meghatározott célállomások érintésével a kiindulási pontba tér vissza?
- Változik-e ez az út, és hogyan, ha villamos vagy troli használata is megengedett?

Elvárásunk tehát hogy az ilyen típusú kérdésekben a használható tömegközlekedési eszközök köre szabadon megválasztható legyen.

Először az elméleti modellt alkotó entitások körét és az entitások attribútumait határoztam meg. Ennek alapján került sor az adatmodell Autodesk Map 6 környezetben való felépítésére. Az objektumok definiálását, az attribútumok megadását és magának a hálózat topológia

megalkotását igen részletesen ismertettem. Ezt az indokolta, hogy az eredményeket szemléltető alkalmazás úgy lett felépítve, hogy annak adatbázisa szabadon lecserélhető, amennyiben az útmutatásokat pontosan követtük.

Az alkalmazás legfontosabb párbeszédpaneljét kiszolgáló adatstruktúrák és függvények egyéni megoldásokat is tartalmaznak. Például egyetlen lista típusú programváltozó segítségével valósítottam meg az egyes elemek két lista doboz közötti mozgásának nyomon követését. E struktúra egyik előnyeként említhető, hogy mindkét lista rendezettsége automatikusan biztosított.

A célként kitűzött leggyorsabb útvonal meghatározásához az összes, közlekedéssel kapcsolatos fólián található vonalláncként definiált objektumból egyetlen hálózat topológiát definiáltunk. A párbeszédpanelben található, a tömegközlekedési eszköz típusok használatát engedélyező három jelölőnégyzet kitöltésétől függően az útvonal meghatározáshoz igénybe vehető élek halmaza viszont változó. Ezt az ellentmondásos helyzetet oldja fel a résztopológia definíciójának bevezetése.

Az új fogalom gyakorlati megvalósításaként bemutatom azt a megoldást, amellyel biztosítható, hogy a projektben mindig az elemzések végrehajtásához szükséges helyes résztopológiát alkotó objektumok legyenek jelen. A hálózat elemzés eredményeként előállított információ nem csak grafikusan jelenik meg a rajzterületen, hanem riportszerű szöveges állomány is készíthető, mely a képernyőn megjelenítve vagy nyomtatott formában is elérhető. Emellett egy vázlatos térkép is nyomtatható az ajánlott útvonalról, ahol az utcákat körvonalaik képviselik, az útvonal megjelenítéshez alkalmazott vonaltípusok pedig a nem színes nyomtatón készült térképek esetén is megkönnyítik a tömegközlekedési eszközök típusainak azonosítását.

Egy utca térképen való pontos elhelyezkedésének meghatározásához egy felhasználóbarát funkcióval egészítettem ki az alkalmazást. A listából kiválasztott utca kijelölését követően a rajz nézete a láthatóságra optimalizálva kerül beállításra.

A CD mellékleten található Visual LISP nyelven kifejlesztett alkalmazás jelenleg két nyelven is tud kommunikálni a felhasználóval. A nyelv kiválasztásához külön eszköztárat definiáltam, mely segítségével bármikor megváltoztathatjuk az alkalmazás nyelvét. Ráadásul mindezt olyan

formában valósítottam meg, hogy a részletezett útmutatást követve a használható nyelvek köre utólag is, tetszőlegesen tovább bővíthető, vagy akár szűkíthető.

Külön fejezetben ismertetem egy korábbi eredményemet, mellyel a *pont a poligonban* algoritmust sikerült általánosítani. Az algoritmust felhasználtam az alkalmazásban is, ezért annak teljes Visual LISP kódja megtalálható az értekezés CD mellékletén.

Végül bemutatok néhány általam definiált AutoCAD parancsot, melyekre az alkalmazásban többször is szükségem volt, s melyek az alkalmazástól eltekintve is jól használhatóak akár Autodesk Map akár csupán AutoCAD felületen.

9. Summary

The results of this dissertation can be grouped around a single central theme, which is the solution of the problems arisen during the modelling and analysing of urban public transport in Autodesk Map.

The dissertation begins with the exact determination of the field of research. The topologic data model is suitable for execution of the most varied data analysis from the vector based geographic information systems. The Autodesk Map is exactly a software product of this sort based on topologic model that means the review of the program's tools covers the details about topologic model.

I give a critical summary about the network topology, which is stressed from the aspect of the results, and its analysing tools.

I make use of Autodesk Map's customisation tools in order to demonstrate my results in practice that is why I review the most important information about defining custom menus, dialogue boxes and the application developing interface called Visual LISP.

I have worked out and now introduce a new concept of network part-topology which meets the claim of analysing inhomogeneous topologies. The new concept can be demonstrated perfectly by a model of urban public transport. In the model a topology of network type can be defined which can be analysed while looking for the answers to the questions of following types:

- Which is the fastest route if for example the use of bus is allowed, which starts from a given point, passes one or more visit points determined in advance, and finally turns back to the starting point?
- Does this route change, and how if the use of tram is permitted also?

Our demands are that the type of public transport could be selected without restriction in questions of this kind.

First I determined the entity types building the theoretical model and their attributes. On the grounds of this the data model was built in the environment of Autodesk Map Release 6. The definition of objects, the realisation of attributes and the building of the network topology are

described in detail. The reason for this is that the application demonstrating my results was built in the following way: Keeping the instructions strictly, the graphical database can be exchanged.

The data structures and functions serving the most important dialogue box of the application contain some custom solutions also. For example I realised the recording the moves of items between the two list boxes with the help of a single program variable of list type. One of this structure's advantages is that the sort of both lists is created automatically.

A single network topology is defined from all the objects of polyline situated on the layers related to transport in order to determine the fastest route, which was set as an aim. There are three checkboxes controlling the available types of public transport in the dialogue box. Their fillings control the set of links, which can be used to determine the best route that is to say the set of links is always varies. This contradiction can be solved by introducing the concept of part-topology.

To realise the new concept in practice I demonstrate a procedure, which ensures that the objects belonging to the correct part-topology required to applying the analysis would be copied into the project. The information produced by the network analysis appears not only graphically on the drawing area, but a text file (report) can be created, which can be read on the screen or can be printed. Besides a roughly outlined map can be printed also, on which the streets are indicated by their outlines, while the applied line types used to denote the route itself make easier to recognise the type of public transport taken during the route.

I completed the application with a user-friendly function as well that helps to determine the exact location of a street. After selecting the name of the street from a list, the corresponding objects become highlighted, and the view of the drawing changed depending on the objects' visibility.

The application developed in programming language called Visual LISP, and can be found on the CD enclosed to the dissertations communicates with the users in two languages (Hungarian and English) at the moment. To change the language I defined a separate toolbar, which helps to modify the language of the application. Following the instructions detailed in the dissertation, the scope of the available languages can be widen or restricted arbitrary.

I discuss my former results in a separate chapter, which is about the generalisation of the algorithm called *point in polygon*. This algorithm was used by the applications as well, so its entire Visual LISP source code can be found on the CD enclosed to the dissertation.

Finally I reviewed some AutoCAD command developed by myself, which I needed in the application several times, and which can be applied well not even in applications but in the interface of either Autodesk Map or AutoCAD.

Irodalomjegyzék

[1] AutoCAD Developer Documentation, Autodesk, Inc. 2002.

[2] AutoCAD Map 2000 Felhasználói Kézikönyv, Autodesk, Inc. 2000.

[3] Autodesk Map Getting Started, Autodesk, Inc. 2002.

[4] Autodesk Map Visual LISP/ADSRX API, Autodesk, Inc. 2002.

[5] BALLA-TÓTH-TRIPSÁNSZKY-NÉMETH: Térinformatikai szakági programozás, Műegyetemi Kiadó, 1995.

[6] BARTELME, N.: Geoinformatik, Springer, 1995.

[7] DAVID MARTIN: Geographic Information Systems, Routledge,1996.

[8] DETREKŐI-SZABÓ: Térinformatika, Nemzeti Tankönyvkiadó, 2003.

[9] KERTÉSZ ÁDÁM: A térinformatika és alkalmazásai, Holnap Kiadó, 1997.

[10] TIKÁSZ-KRAUTER-UGRIN-CSORNAI: A digitális térkép geometriai alapjai, Műegyetemi Kiadó 1995.

[11] KOLLÁNYI–PRAJCZER: Térinformatika a gyakorlatban, GeoGroup Bt. 1995.

[12] MACZKÓ-NAGY: LISP, AutoLISP programozás AutoCAD-ben IBM PC-n, LSI 1989.

[13] NCGIA Core Curriculum Bevezetés a térinformatikába, (Szerk. Márkus B. Goodchild, M.F.–Kemp, K.K. nyomán), Székesfehérvár, EFE FFFK 1994.

[14] NCGIA Core Curriculum Térinformatikai alapismeretek, (Szerk. Márkus B. Goodchild, M.F.–Kemp, K.K. nyomán) Székesfehérvár, EFE FFFK 1994.

[15] NCGIA Core Curriculum Térinformatikai alkalmazások, (Szerk. Márkus B. Goodchild, M.F.–Kemp, K.K. nyomán) Székesfehérvár, EFE FFFK 1994.

[16] NCGIA Core Curriculum Térinformatika Magyarországon, (Szerk. Márkus B.) Székesfehérvár, EFE FFFK 1994.

[17] NCGIA Core Curriculum Térinformatikai Értelmező Szótár, (Szerk. Márkus B.) Székesfehérvár, EFE FFFK 1996.

[18] STEIN, DAVID M.: The Visual LISP Developers Bible, 2003. (www.dsxcad.com/book)

[19] ZICHAR M.: Operations of Topology in GIS, *Proceedings of Seminar "Geometry and Graphics in Teaching Contemporary Engineer",* Wisla 2000.

[20] ZICHAR M.: Overlay Operations in GIS, *Proceedings of the 4th International Conference on Applied Informatics, Eger-Noszvaj*, 1999. Pages 217-222

[21] ZICHAR M.: GIS and Computer Graphics, *Geometria i grafika inzynierska z.2*, (*Journal of Silesan Institute of Technology*) *Gliwice* 1998. Pages 35-43

Függelék

Publikációk listája

1. M. Zichar: GIS in Practice, *Scientific Journal of Silesian Technical University* (megjelenés alatt)
2. M. Zichar: Operations of Topology in GIS, *Proceedings of Seminar "Geometry and Graphics in Teaching Contemporary Engineer",* Wisla 2000.
3. M. Zichar: Overlay Operations in GIS, *Proceedings of the 4th International Conference on Applied Informatics, Eger-Noszvaj*, 1999. Pages 217-222
4. M. Zichar: GIS and Computer Graphics, *Geometria i grafika inzynierska z.2 (Journal of Silesan Institute of Technology), Gliwice* 1998. Pages 35-43
5. M. Zichar: A tutorial program, *Proceedings of the 2nd International Conference on Applied Informatics, Eger*, 1995. Pages 165-169

Egyetemi oktatási segédanyag

1. Zichar M: Oktatási segédlet az AutoCAD Map 2000 programhoz, 2000. (66 oldal)