3D városmodell kialakítása és megjelenítése PDA eszközökön

Kottyán László
Nyugat-magyarországi Egyetem Geoinformatikai Kar

ÖSSZEFOGLALÁS

A 3D Városkalauz projekt célkitűzése a Székesfehérvár belvárosában meghatározott mintaterületről nyert geometriai adatok és a kapcsolódó turisztikai, régészeti adatok egységes információs modellbe integrálása; valamint, a modell alapján, egy turisztikai célú alkalmazás fejlesztése PDA eszközökre.
A cikk ismerteti az adatok feldolgozásának módszerét és a 3D modell mobil eszközökön történő megjelenítésének vizsgálatát.

A 3D VÁROSKALAUZ PROJEKT BEMUTATÁSA

A 3D Városkalauz kutatási projekt a Nemzeti Kutatási és Technológiai Hivatal által támogatott projekt, amely a főpályázó, Fehérvár ÉPÍTÉSZ Kft és a konzorciumi partner, GEOINFO Nonprofit Kft. összefogásával valósul meg. A kétéves projekt befejezési időpontja 2011. április 30-a.

A projekt célja, a napjaink innovatív technológiái, módszerei alkalmazásával egyrészt, a valós 3D modellek előállításának és mobil eszközökön történő megjelenítésének vizsgálata turisztikai célú felhasználásra; másrészt, egy mintarendszer elkészítése, amely módszereket, eljárásokat kínál turisztikai információs rendszerek kialakításához PDA eszközökön.

A projekt résztvevői a 3D városmodell kialakításához a CityGML nyílt adatmodell alkalmazása mellett döntöttek [4]. A CityGML adatmodellt, a földrajzi adatokat leíró nyílt OGC szabvány, a GML 3 (Geography Markup Language) alkalmazási sémájaként implementálták. Célja, a városi objektumok reprezentálása olyan módon, hogy alkalmas legyen virtuális 3D modellek tárolására. 2008. augusztus 20-án az Open Geospatial Consortium a CityGML 1.0.0 verzióját hivatalos OGC szabványként fogadta el.

A CityGML, ellentétben a GML modellel, az objektumok geometriája mellett, azok tematikus jellemzőit is kezeli. A nyelv geometriai modellje a 3D városmodellek geometriáját és topológiáját, míg a tematikus modell az objektumok szemantikus információit reprezentálja. A szemantikus

modell, a geometriához jelentéseket társítva, lehetővé teszi az objektumok közötti kapcsolatok meghatározását. Így, ugyanaz a 3D modell felhasználható különböző alkalmazási területeken, az un. szemantika-vezérelt megjelenítés révén [3] [5].

A cikk bemutatja a CityGML modell előállításának egy lehetséges módszerét.

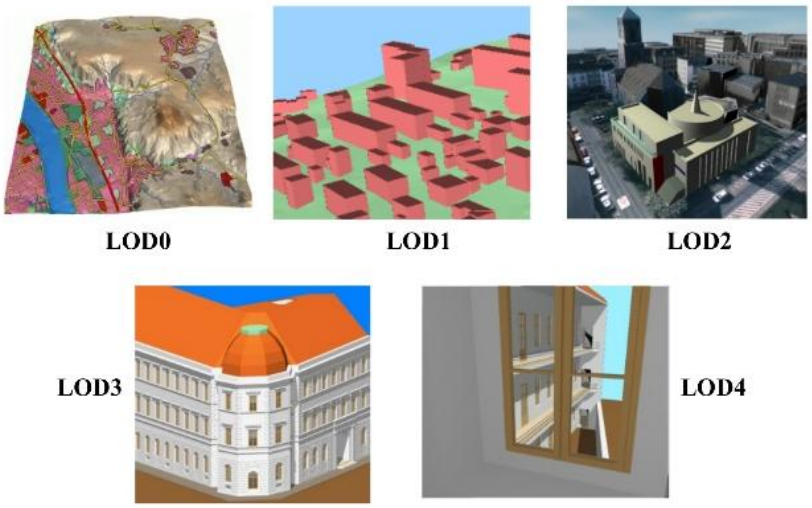
A CITYGML ADATMODELL KIALAKÍTÁSA

A CityGML moduláris felépítésű, egy adatmodell több modul felhasználásával alakítható ki. A modulok a következők:

- Core: a modell alapvető beállításait tartalmazza, és hivatkozásokat a további felhasználandó tematikus modulokra.
- Apperance: a modullal a modell objektumaihoz megjelenítési információk társíthatók.
- Building: az épületek, épületrészek, és az épületek belső részeinek modellezésére alkalmas modul, az objektumok többszintű részletességgel definiálhatók (1. ábra).
- CityFurniture: nem mozdítható objektumok modellezésére alkalmas modul, mint például az utcalámpák, közlekedési táblák, hirdetőtáblák, utcai padok.
- CityObjectGroup: a modell tetszőleges objektumai csoportokba foglalhatók a modul segítségével.
- Generics: a modul egy kiterjesztési lehetőséget biztosít, amellyel olyan attribútumok, osztályok definiálhatók, amelyek nem találhatók meg a tematikus modulokban.
- LandUse: a modul egy földterület földhasználati adatainak reprezentálására szolgál.
- Relief: a modul a domborzat meghatározását teszi lehetővé.
- Transportation: utak, vasútak, terek modellezésére.
- Vegetation: a növényzet modellezésére, a modell objektumai lehetnek egyedi növények (pl. fa) vagy növényzettel borított területek (pl. erdő).
- WaterBody: folyók, csatornák, tavak, vízgyűjtők modellezésére alkalmas modul.
- TexturedSurface: a modul az objektumok felületinek vizuális reprezentációjára alkalmas. A szabvány későbbi verzióiban ez a modul felfüggesztésre kerül, ezért helyette az Apperance modul használatát javasolják.

A moduláris felépítés lehetőséget biztosít, arra, hogy az alkalmazásokhoz kialakítandó adatmodellben a modulok igény szerinti kombinációja legyen felhasználható. A modulok egyfajta összeállítása a profil. A szabványban definiált összes modul által alkotott profil az un. *base profile*.

Az olyan alkalmazás specifikus adatok, amelyek a szabványban nem biztosítottak a Generics modulban definiálhatók vagy használható az un. ADE (Application Domain Extension) mechanizmus. Az ADE kiterjesztés külön XML sémaként adható meg, definiálva az alkalmazáshoz szükséges elemeket [5].



0-1. ábra Objektumok megjelenítése különböző részletességben [5]

A mintarendszer funkciói és speciális adatai

A mintarendszer GPS vevővel rendelkező mobil eszközön kerül telepítésre. A rendszer fő funkciója a turisták kalauzolása látnivalótól-látnivalóig és a 3D megjelenítés biztosítása. A kalauzolás tervezett megoldása a hang alapú és a szöveg alapú tájékoztatás a magyar mellett, angol, német és francia nyelveken.

A rendszer fontos jellemzője az idő dimenzió, ugyanis egy objektum történelmi vonatkozású leíró és térbeli adatokkal is rendelkezhet. A mintarendszer 3D objektumai az alábbiak szerint csoportosíthatók:

- városi objektumok, amelyek jelenleg láthatók (épületek, szobrok, utcai objektumok),
- az épületeken belüli objektumok (pl.: történelmi, kulturális vagy művészettörténeti jelentőségű belső terek, festmények),
- történelmi objektumok, amelyek már nem láthatók, de régészeti és művészettörténeti kutatások eredményeként modellezhetők (pl.: lerombolt épületek, várfalak).

A mintarendszer tehát a szabványban meghatározott elemeken túl további adatokat is tartalmaz, ehhez a szabvány kiterjesztése szükséges. A kiterjesztés definiálja:

- a történeti és művészettörténeti leírásokat,
- az objektumokhoz tartozó képi (pl. fénykép, alaprajz,) tartalmak jellemzőit,
- a turisztikai információkat (pl. kalauzolás több nyelven, hanganyagok),
- az idő dimenzió adatait.

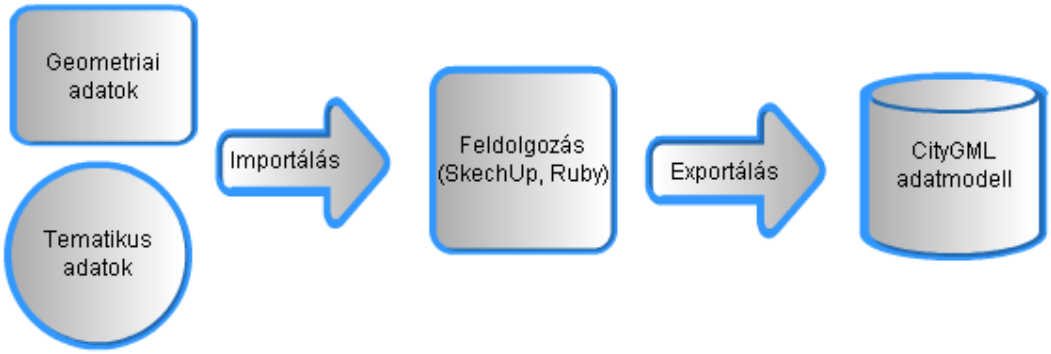
A modellezés folyamata

A projekt korábbi feladatainak teljesítésének eredményeként rendelkezésre állnak a mintaterület geometriai adatai és tematikus adatai [4].

A geometriai adatok és a tematikus adatok egy adatmodellbe integrálásához a Google SketchUp szoftvert használtuk fel. Google SketchUp egy modellező szoftver, amely rendelkezik egy Ruby alkalmazásprogramozói felülettel. Ezért, a Ruby API felhasználásával a SketchUp alkalmazás kiterjeszthető, saját készítésű modulokkal [2].

A CityGML adatmodell kialakításához a projekt keretében elkészült egy importáló és egy tematikus adatok kezelésére alkalmas szkript, valamint felhasználtuk a CityGML-Toolchain Editor 1.4 plugint[1].

Az eljárás folyamatát a 2. ábra szemlélteti.



2. ábra CityGML adatmodell előállítása

Az eljárás lehetővé teszi az importált geometria pontosítását, kiegészítését. A Google 3D Warehouse szolgáltatással elérhetők olyan objektumok, amelyekkel a virtuális környezet utcai elemei (fák, padok, lámpaoszlopok) modellezhetők. További előnye a megoldásnak a textúrázás lehetősége.

A MEGJELENÍTÉS VIZSGÁLATA

A három dimenziós modell mobil eszközön történő megjelenítését a mobil eszköz teljesítménye nagy mértékben meghatározza. Elsősorban a processzor kapacitása és a rendelkezésre álló memória felhasználása jelent korlátokat a modell megjelenítése, kezelhetősége szempontjából.

Felhasználói szemszögből nézve, egy virtuális három dimenziós környezet kezelésekor azt várhatjuk el, hogy a felhasználói műveletek a felhasználói élményt támogassák egyfajta folyamatosság látszatát keltve a modell kirajzolásakor.

A felhasználói alapműveletek a következők:

- modell kicsinyítése és nagyítása,
- modell eltolása,
- modell forgatása.

A megjelenítés vizsgálata során arra kerestük a választ, hogy a mobil eszköz képernyőjén milyen méretű modell jeleníthető meg és kezelhető a felhasználói elvárásoknak megfelelően. A felhasználói elvárás és a felhasználói élmény minősége szubjektív fogalom, ezért a vizsgálat egy teszt program végrehajtásának időtartamát mérte különböző pontszámú modellek estében. A program végrehajtásának időtartama és a tesztelő személy értékelése alapján állapítottuk meg azt, hogy adott hardver kapacitás figyelembe vételével, adott futtató környezetben, meghatározott adatkezelés mellett egyidejűleg mennyi pont kezelése javasolt.

Az előzetes vizsgálatok alapján a tesztelő algoritmust optimalizáltuk, majd a végeredmények alapján következtetéseket vontunk le a modell műveleteinek implementálásával kapcsolatban.

Tesztelési környezet

A tesztekhez felhasznált mobil eszköz fontosabb paraméterei:

- Típus: Mio P560,
- Operációs rendszer: Windows Mobile 6.0 classic,
- Kijelző:
  - mérete: 3.5",
  - felbontása: 320x240,
- Memória:
  - belső ROM: 64 MB, telepített .NET CF 2.0-val,
  - belső RAM: 2GB.

A kódolást a Basic4ppc IDE használatával valósítottuk meg, amely natív, a .NET CF keretrendszer által végrehajtható állományt állít elő.

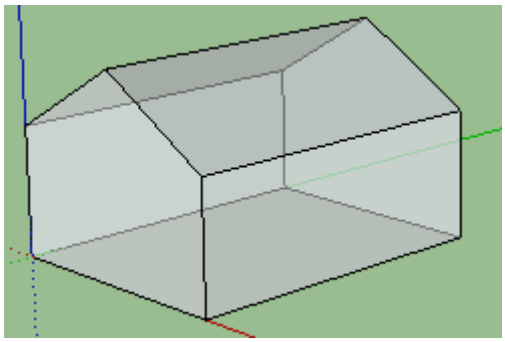
A 3D adatok kezelése

A vizsgálathoz négy geometriai modellt használtunk fel (1. táblázat), amelyeket a Google SketchUp programban állítottunk elő.

1. táblázat: Teszt modellek

Modell azonosító	Épületek száma	Pontok száma	Poligonok száma
B1	1	10	7
B10	10	100	70
B100	100	1000	700
B1000	1000	1000	7000

A modellben épületeket ábrázoltunk, egy épületet tíz ponttal és hét poligonnal adtunk meg (2. ábra).



2. ábra B1 modell

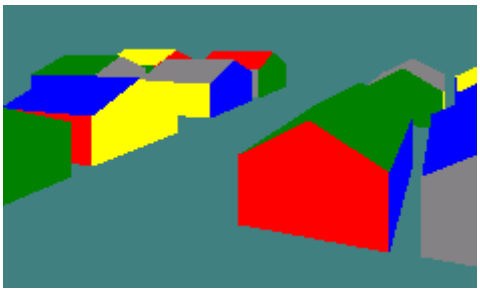
A modelleket a teszt programban egy-egy objektumként dolgoztuk fel, amelyhez a SketchUp modellt átalakítottuk, külön szöveges állományban tárolva a pontok x,y,z koordinátáit és a poligonokat alkotó pontok azonosítóit.

A program első lépésként beolvasta és tömbökben tárolta az állományok tartalmát. Ezt követően minden egyes poligonhoz egy színkódot rendelt, amelyet a színek tömbjében tárolt. Egy modell egy háromdimenziós objektumban került feldolgozásra (3. ábra).

A három dimenziós objektum a modellről a következő adatokat tartalmazta:

- nV - pontok száma,
- n P - poligonok száma,
- ver() - hivatkozás a pontok tömbjére,
- pol() - hivatkozás a poligonok tömbjére,
- col() - hivatkozás a színek tömbjére,
- posX, posY, posZ - a modell középpontjának koordinátái,
- angleX, az, angleY, angleZ - a modell x, y, z irányú forgásszöge.

A háromdimenziós objektum további jellemzői a raszteres képi háttér és a felbontás. A felbontás meghatározza a képi koordináta-rendszer és a modell koordináta-rendszere közötti viszonyítási arányt.



3. ábra B10 modell forgatás közben

A vizsgálat algoritmusa

A teszt program célja a felhasználói műveletek emulálása, mérhető paraméterek felhasználásával, amely alapján az egyes teszt esetek összehasonlíthatók és következtetések vonhatók le a mintarendszer elkészítésére vonatkozóan. A teszt program rögzítette a végrehajtásra fordított időtartamot másodpercekben és az egyes műveletek memória felhasználását.

Az algoritmus a következő lépésekből állt:
1. ReadCloud() - pontok beolvasása állományból, majd tárolása tömbben
2. ReadPol() - poligonok beolvasása állományból, színek hozzárendelése, majd a poligonok és a színek tárolása tömbökben
3. Teszt futtatása
   3.1. Rot() - modell forgatása az y tengely körül az óra járásával megegyező irányba
   3.2. Mov() - modell eltolása: x + 50, y + 50, x - 50, y - 50 értékekkel
   3.3. Rot() - modell forgatása az y tengely körül az óra járásával ellentétes irányba
   3.4. Mov() - modell eltolása: x - 50, y - 50, x + 50, y + 50 értékekkel
   3.5. Zoom() - kicsinyítés (scale 5-től 1-ig), nagyítás (scale 1-től 10-ig), kicsinyítés (scale 10-től 5-ig)

Az algoritmus optimalizálása

Az algoritmusban definiált műveletek megvalósításának módja hatással van a megjelenítésre. A koordináta értékekkel végzett műveletek estében a pontok nagy száma jelentős processzor kapacitást igényelhet, amelyet a felhasználó úgy érzékel, hogy lassan rajzolódik ki a képernyőn a három dimenziós objektum vagy szaggatottan jelennek meg a modell egyes részei eltolás, forgatás közben. A pontok száma a memóriában tárolt objektum méretét is meghatározza. Amennyiben a felhasználható memória nem elegendő memória felszabadítására van szükség, amely történhet a futtató környezet által automatikusan, vagy a programozó által explicit módon, a forráskódban megadva.

Az algoritmus egyes lépéseinek megvalósításakor olyan megoldást kerestünk, amely nagy pontszámú modell mellett is biztosítja a felhasználó számára megfelelő megjelenítést. Az algoritmus implementálása és a tesztelés iteratív és inkrementális tevékenységként jellemezhető. A számos megoldásból az elsőt A változatnak, az utolsó, optimálisnak elfogadottat B változatnak jelöltük meg. Asztali számítógépes környezetben mindkettő implementáció, a B1000 modellen futatva is, megfelelő eredményt hozott felhasználói szemszögből. A megoldások között tehát a mobil környezetben, korlátozott erőforrások mellett történő felhasználhatóság jelenti a különbséget.

Tekintsük át az egyes megoldások közötti különbségeket.

Az A változat műveleteinek megvalósítása

Az A változat, a korábban említett raszteres háttéren jeleníti meg a modellt. A felhasználói műveletek kettő koordináta-rendszerben értelmezhetők.

<u>Az objektum létrehozása</u>

A 3D objektum létrehozásakor meghatározzuk a raszteres háttér méretét és a felbontás arányát. Abban az esetben, ha kép mérete 200x200 pixel és a felbontás aránya 1, akkor a koordináta-rendszer beosztása 100 egység minden irányban.

A modell megjelenítése (egyszerűsített jelöléssel):
draw(ver(), pol(), col(), posX, posY, posZ, angleX, angleY, angleZ)

A forgatás, implementálása:
```
Rot( y)
  updated = true
  for i= 1 to 360
        draw(ver(), pol(), col(), 0, 0, 0, 0, y, 0)
  end
```

Ahol, y egy konstans érték, a forgatás inkrementális, a pontok koordinátáinak értéke frissítésre kerül minden egyes forgatás esetén.

Az eltolás implementálása:
```
Mov(delta)
 for i = 1 to 50
        posX = posX + delta
        draw(ver(), pol(), col(), posX, 0, 0, 0, 0, 0)
 end
```

Ahol, delta a modell x irányú léptetésének értéke.

A nagyítás implementálása:
```
Zoom(s)
  scale = s
  draw(ver(), pol(), col(), 0, 0, 0, 0, 0, 0)
```

Az átadott paraméter az aktuális felbontást mértékét adja meg.

Az B változat műveleteinek megvalósítása

Az objektum létrehozása
Az objektum létrehozása nem változik. Azonban ennél a változatnál megkülönböztetjük a raszteres hátteret és a képernyő hátterét. Amíg az A változatnál a háttérkép a teljes nézetet jelentette, a B változatnál a nézetet az a felhasználói felület elem határozza meg amelyen a kép elhelyezkedik. Ezáltal még egy viszonyítási rendszert alkalmazunk, amelyen a kép pozícióját állíthatjuk be.

A forgatás, implementálása:
```
Rot(y)
  updated = false
```

```
for i= 1 to 360 step y
        draw(ver(), pol(), col(), 0, 0, 0, 0, i, 0)
end
```

Ahol, y a forgatás mértéke, a pontok koordinátáinak értéke nem változik.

Az eltolás implementálása:

```
Mov(delta)
 for i = 1 to delta
        image.left +i
 end
```

Ebben az esetben a raszteres kép eltolása történik meg egy raszteres művelettel.

A nagyítás implementálása
Megegyezik az A változatnál bemutatottakkal.

További optimalizálási lépések:
Bár a .Net keretrendszer automatikusan kezeli a memóriát, a tesztek kimutatták, hogy egyes műveleteknél az explicit memória felszabadítással a megjelenítés hatékonysága növelhető volt.

A tapasztalatok azt mutatták, hogy bármelyik alkalmazott algoritmus esetén, érdemes a perspektív megjelenítést egyedileg beállítani, a felbontástól és a modell kiterjedésétől függően.

A vizsgálat értékelése

A 2. táblázat az A és a B implementációk futtatási eredményeit mutatja a B1, B10, B100, B1000 modellek esetén.

A táblázatban a felhasználói élmény értékelése megfelelő és elfogadhatatlan minősítésekkel történt, amely a tesztelő személy véleményét mutatja. A megfelelő minősítés a modell folyamatos kirajzolását jelenti. Az elfogadhatatlan esetekben az animáció sebessége nem tenné lehetővé a kívánt felhasználást.

A teszt eredmények tükrében megállapítható, hogy az optimalizált algoritmussal ezer pontot és hétezer poligont tartalmazó modell is megfelelően kezelhető egy nézetben.

2. táblázat A tesztek eredményei

Modell azonosítója	Paraméterek	A teszt	B teszt
B1	végrehajtás időtartama (s)	21	6
	kezdeti memória igény (B)	10 184	10 168
	max. memória felhasználás (B)	561 636	14 400
	felhasználói élmény	megfelelő	megfelelő
B10	végrehajtás időtartama (s)	49	9
	kezdeti memória igény (B)	13 076	13 060
	max. memória felhasználás (B)	901 688	20 132
	felhasználói élmény	megfelelő	megfelelő
B100	végrehajtás időtartama (s)	332	30
	kezdeti memória igény (B)	41 904	41 888
	max. memória felhasználás (B)	668 800	83 140
	felhasználói élmény	elfogadhatatlan	megfelelő
B1000	végrehajtás időtartama (s)	3273	227
	kezdeti memória igény (B)	329 888	329 920
	max. memória felhasználás (B)	1 913 884	763 572
	felhasználói élmény	elfogadhatatlan	elfogadhatatlan

A FEJLESZTÉS FELADATAI

A cikk készítésekor a projekt résztvevői a mintarendszer fejlesztési feladataival foglalkoznak a megjelenítés vizsgálatával kapcsolatos tapasztalatokat felhasználva. A megjelenítés vizsgálata Windows Mobile 6.0 operációs rendszert futtató PDA készüléken történt, azonban a tesztek tapasztalatai általánosíthatók.

A megjelenítés megvalósításánál elsősorban a CityGML LOD2 részletességi szintjének megfelelően kerül kialakításra a modell, azonban egyes régészetileg, művészettörténetileg indokolt esetekben a LOD3 szinttű megjelenítést kívánjuk alkalmazni.

A fejlesztés kiemelt feladatai közé tartozik a CityGML adatok kezelésének megvalósítása a mobil eszközön, a GPS jelek feldolgozása és a tematikus adatok megjelenítése.

IRODALOM

1. CityGML-Toolchan Editor 1.4, http://www.citygml.de/index.php/sketchup-citygml-plugin.html
2. Google SketchUp Ruby API, http://code.google.com/intl/hu/apis/sketchup/
3. Kolbe, T.H.: CityGML Home, http://www.citygml.org/
4. Kottyán L.: Adatmodellezés CityGML használatával, GISopen 2010, NymE Geoinformatikai Kar, Székesfehérvár, 2010
5. OGC: OpenGIS® City Geography Markup Language (CityGML) Encoding Standard, OGC 08-007r1, 2008

A szerző elérési adatai

Kottyán László
Nyugat-magyarországi Egyetem
Geoinformatikai Kar
8000 Székesfehérvár
Pirosalma u. 1-3.
Tel. +36 22 516 553
Email: kl@geo.info.hu
Honlap: www.geo.info.hu