Python OGR modul (vektoros adatok kezelése)

Szél Henrietta

Az adatforráshoz való hozzáférés

A nyílt GDAL/OGR (Geospatial Data Abstraction Library, OpenGIS Simple Features Reference Implementation) könyvtárak számos eszközzel segítik a térképészeti adatok feldolgozását. A GDAL a raszteres, az OGR pedig a vektoros adatok kezeléséért felel. A C++ nyelven írt OGR könyvtárhoz úgynevezett Python kötéseket készítettek, hogy Python programokból elérhetők legyenek az OGR funkciói: ez a Python OGR modul.
Az OGR könyvtár segítségével számos vektoros formátumot (állománytípust vagy egyéb adatforrást) tudunk kezelni, például:

- ESRI shapefile
- personal geodatabase (térinformatikai adatokat tároló Microsoft Access adatbázis)
- ArcSDE adatbázis
- MapInfo formátum
- GRASS formátum
- Bentley Systems MicroStation formátum
- TIGER/Line
- SDTS
- GML
- KML
- MySQL, PostgreSQL, MariaDB, stb
- Oracle Spatial
- Informix
- ODBC

A különböző fájltípusoknak és más adatforrásoknak a kezelésére az OGR könyvtár úgynevezett meghajtókat (vagy driver-eket) használ. A következő Python kóddal tudjuk megvizsgálni, hogy milyen driver-ek állnak rendelkezésünkre:

```
from osgeo import ogr
driverList = []

for i in range(ogr.GetDriverCount()):
    driver = ogr.GetDriver(i)
    driverName = driver.GetName()
    if not driverName in driverList:
        formatsList.append(driverName)

for i in formatsList:
    print i
```

A kiírt nevek alapján azonosíthatjuk a megfelelő drivert. A Shape fájl formátumot például az „ESRI Shapefile” nevű meghajtóval kezelhetjük. A megfelelő driver név szerint is elérhető (ha nincs telepíve az adott nevű driver, akkor a GetDriverByName függvény None értéket térít vissza). A Shape fájlokat kezelő meghajtót tehát a következő függvényhívással érhetjük el:

```
driver = ogr.GetDriverByName('ESRI Shapefile')
```

Az állományt ezután a meghatón keresztül nyitjuk meg az Open függvény segítségével, aminek első paramétere az állomány neve (teljes elérési útvonal), a második pedig egy egész szám, aminek értéke 0 vagy 1 lehet. A második paraméter 0, ha az állományt csak olvasásra nyitjuk meg, az érték 1 ha írni is szeretnénk bele.

```
file = driver.Open(filename, 0)
if file is None:
    print ('Nem tudtam megnyitni a fájlt!')
```

Amennyiben a meghajtó nem tudta megnyitni az állományt, akkor a None értéket adja vissza. Ez a helyzet akkor fordulhat elő, ha a Shape fájl tartalma sérült vagy az shx vagy dbf fájl nem található.

A térképészeti adatok kinyerése

A következő lépés a Shape fájlban található réteghez (layer) való hozzáférés. Ezt a funkciót a GetLayer(index) függvény biztosítja. Shape fájlok esetében az index mindig 0 (vagy el is lehet hagyni ezt a paramétert), az index csak olyan formátumok esetében hasznos, mint pl. a GML vagy a TIGER. A következő sorral tehát a Shape fájl egyetlen rétegét szerezzük be:

```
layer = datasource.GetLayer()
```

Ezután következik a rétegen található elemek (features) beolvasása. A feature-ek számát a layer GetFeatureCount() függvényével kérhetjük le, és az egyes feature-ek a GetFeature(index) függvénnyel érhetjük el. Vagy végig lehet menni az összes feature-en a következő kóddal:

```
feature = layer.GetNextFeature()
while feature:
    # feldolgozás
    feature = layer.GetNextFeature()
layer.ResetReading() #ha újra kell kezdeni a beolvasást
```

A Shape fájlunk csak egyetlen feature-t tartalmaz, ezért a GetNextFeauter() egyszeri meghívásával megoldjuk a hozzáférést.
Az elem mértani objetkumát a GetGeometryRef() függvénnyel kérhetjük le, típusát pedig a GetGeometryType() vagy GetGeometryName() függvénnyekkel ellenőrizhetjük le. A következő sorokban például azt ellenőrizzük le, hogy a poligon vagy multipoligon típusú elemmel van-e dolgunk:

```
geometry = feature.GetGeometryRef()
if geometry.GetGeometryName() == 'POLYGON' or geom.GetGeometryName()
=='MULTIPOLYGON':
    # feldolgozás
```

A típusokat az OGR konstansaival (pl. ogr.wkbPoint, ogr.wkbLineString, ogr.wkbPolygon, ogr.wkbMultiPoint, ogr.wkbMultiLineString, ogr.wkbMultiPolygon, stb.) is azonosíthatjuk.

Attribútumok

Az attribútomokat a GetField() függvénnyel és annak variációival érhetjük el. Például:

```
attr = feature.GetField('id')
```

```
attrstr = feature.GetFieldAsString('id')
```

Az attribútumok számát a GetFieldCount() függvénnyel kapjuk meg.

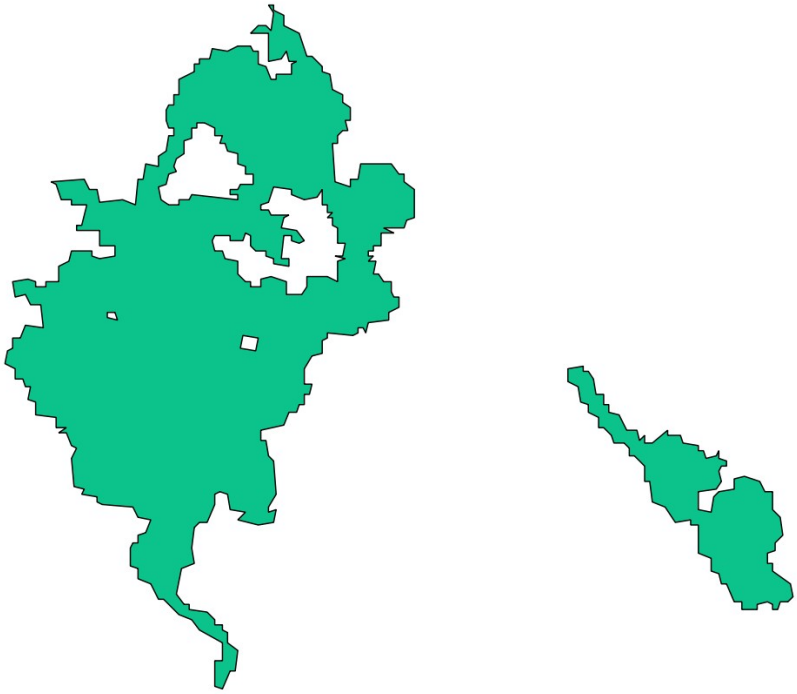
Geometria egyszerűsítése

A következő Python kód egy Shape file geometriáját egyszerűsíti, adott toleranciával. A fontosabb sorokhoz magyarázatot fűztünk.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import os, sys
from osgeo import ogr

# infile - bemeneti állomány neve
# infile - kimeneti állomány neve
# tolerance - a egyszerűsítés paramétere

def simplify(infile, outfile, tolerance):

    # az ESRI Shapefile meghajtó
    driver = ogr.GetDriverByName('ESRI Shapefile')

    # olvasásra nyitjuk meg a bemeneti állományt
    infile = driver.Open(infile,0)

    if infile is None:
        print 'Nem tudom megnyitni a(z) ', infile, ' nevű állományt!'
        sys.exit(1)

    # a bemeneti állomány adatai: layer, feature, geometry
    inputLayer = infile.GetLayer()
    inputFeature = inputLayer.GetNextFeature()
    geom = inputFeature.GetGeometryRef()
    geomType = geom.GetGeometryType()

    # a kimeneti állomány létrehozása
    if os.path.exists(outfile):
        os.remove(outfile)
    try:
        output = driver.CreateDataSource(outfile)
    except:
        print 'Nem tudom létrehozni a(z)', outfile, ' nevű állományt!'
        sys.exit(1)

    # réteg létrehozása a kimeneten
    outputLayer =
output.CreateLayer('Tolerance',geom_type=geomType,srs=inputLayer.GetSpatialRef()
)
    if outputLayer is None:
        print 'Nem tudom lérehozni a megfelelő layer-t a kimeneti állományban!'
        sys.exit(1)

    outputLayerDef = outputLayer.GetLayerDefn()
```

```
    featureID = 0

    # végigmegyünk az összes elemen
    while inputFeature:

        # az eredeti geometria
        geometry = inputFeature.GetGeometryRef()
        # az egyszerűsített geometria
        simplifiedGeom = geometry.Simplify(tolerance)

        # megpróbálunk létrehozni egy új feature-t az egyszerűsített
geometriával
        try:
            newFeature = ogr.Feature(outputLayerDef)
            newFeature.SetGeometry(simplifiedGeom)
            newFeature.SetFID(featureID)
            outputLayer.CreateFeature(newFeature)
        except:
            print "Nem tudtam létrehozni az egyszerűsített geomatriát!"

        newFeature.Destroy()
        inputFeature.Destroy()
        inputFeature = inputLayer.GetNextFeature()
        featureID += 1


    infile.Destroy()
    output.Destroy()
    print "Az egyszerűsítést sikeresen végrehajtottam"
    return

# a parancssorban átadott paraméterek
if (len(sys.argv) < 4):
    print ('Használat: python simplify.py <bemenet> <kimenet> <tolerancia>')
    sys.exit(0)

# az egyszerűsítő függvény meghívása
simplify(sys.argv[1], sys.argv[2], float(sys.argv[3]))
```

Eredeti geometria:



Egyszerűsített geometria: