

QGIS Python modul készítés¹

QGIS 2.x

dr. Siki Zoltán

Saját modul készítése a 'Plugin Builder' modullal

Egy olyan modult elkészítésén vezetjük végig, mely egy felületeket tartalmazó réteg centrálist tartalmazó pont réteget hozza létre. Ilyen funkciót tartalmaz az *Ftools* modul, de az a centrálist a töréspontok súlypontjában helyezi el, mely nem feltétlenül esik a felületbe konkrét alakzatok esetén. Az általunk elkészített modul garantáltan a felületbe eső centrálist ad.

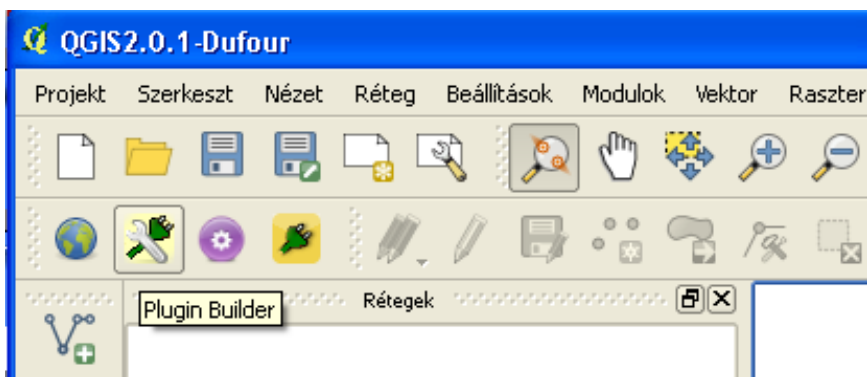
A modul elkészítése során egy szövegszerkesztő programra lesz szükségünk. Használhatjuk a *Jegyzettömböt* (*Notepad*), de ajánlott inkább olyan szövegszerkesztő használat, mely segít a zárójel párok megtalálásában, a kódot szövegét színezve olvashatóbbá teszi azt. Ilyen például a *NotePad++*, *UltraEdit* vagy *Vim*. Még több támogatást kaphatunk a kód elkészítése közben, ha egy integrált fejlesztő környezetet (IDE) használunk, ilyen lehet a *PyScripter* (Windows) vagy a *Spyder* (Linux). A profiknak, nagy rendszerek fejlesztőinek az *Eclipse* fejlesztőkörnyezetet ajánlom, mely Linux és Windows operációs rendszeren is használható.

A munka megkezdése előtt ellenőrizze, hogy a **Plugin Builder** modult telepítette-e. Ezt a Modulok menüben ellenőrizheti. Amennyiben a legördülő menüben látja a *Plugin Builder ...* menüpontot, akkor átugorhatja a következő két bekezdést.

Ha nem látható a menüben a Plugin Builder modul, akkor 1.8 vagy korábbi változat esetén nézze meg a *Modulok/Modulkezelő* menüpontban (2.0 vagy későbbi változatban a *Modul kezelés és telepítés* menüpont *Telepített* fülén), hogy a felsorolásban megjelenik-e a modul. Ebben az esetben kapcsolja be a modult és átugorhatja a következő bekezdést.

Telepítsük a *Plugin Builder* modult. 1.8 vagy korábbi változatokban válassza a *Modulok/Python modulok lekérése* menüpontot, a 2.0 vagy újabb változatban a *Modul kezelés és telepítés* menüpontban a *Továbbiak* fület. A listában válassza ki a Plugin Builder modult és nyomja meg a **Modul telepítés** gombot.

Az eszközsorból válassza a  ikont vagy a *Modulok* menüből a *Plugin Builder*-t.



1 ábra Plugin Builder modul ikonja

A Plugin Builder modul párbeszédablaka jelenik meg, ahol a modulra vonatkozó alapadatokat adhatja meg. A modulnak még nincs magyar változata, ezért angol nyelven jelenik a párbeszédablak tartalma. Az alábbi ábra alapján töltsse ki a párbeszédablakot. Minden egyes Python modul egy új osztály létrehozásával jár, ennek nevét kell az első mezőben megadni. Ezt általában nagy kezdőbetűvel adják meg. A második mezőbe a modul neve kerül, mely egyben a modult tartalmazó fájl neve, amit kis betűvel szoktak írni. A minimális QGIS verzió megadásánál vigyáznunk kell, mert a 2.0 verzióban nem futnak a korábbi verziókra írt modulok, tehát a 2.0 verzióban nem írható ide korábbi verzió.

A 2. ábrán látható adatok beírása után nyomja meg az **OK** gombot. Ezután meg kell adnia a könyvtárat, ahová menteni akarja a modult. Ez egy tetszőleges könyvtár lehet, de a modul használatához a bejelentkezési

¹ Készült a http://www.qgisworkshop.org/html/workshop/plugins_tutorial.html oldal és Carlson Farmer fTools moduljának felhasználásával.

könyvtárában lévő `.qgis2/python/plugins` könyvtárba kell átmásolni a modult, a 2.0 előtti változatoknál a könyvtár a `.qgis/python/plugins` könyvtárba. A mentés után a további teendők jelennek a képernyőn.

QGIS Plugin Builder 2.0.2

Create a template for developing QGIS plugins

Required fields:

- Class Name**
This is the Python class name for your plugin. It should be in CamelCase.
- Plugin name**
This is the name for your plugin that will be displayed in the QGIS plugin manager and the plugin installer
- Description**
A one-liner description of the plugin that appears in the plugin manager and the plugin installer
- Version number**
The version of this plugin
- Minimum required QGIS version**
QGIS version required for this plugin to work
- Text for the menu item**
This is the text that will appear in the menu
- Author/Company name**
Your name or company name (used in the copyright header)
- Email address**
Your email address (used in the copyright header)

Optional fields:

- Bug tracker**
Url of your plugin's bug tracker
- Home page**
Url of your plugin's home page
- Repository**
Url of your plugin repository (if you have one)
- Tags**
A comma separated list of tags that describe the plugin features or function

QGIS Plugin Builder

Class name: RealCentroid
Plugin name: realcentroid
Description: Create point shape of polygon centroids
Version number: 0.1
Minimum QGIS version: 2.0
Text for the menu item: Real centroids
Author/Company: Zoltan Siki
Email address: sik at agt.bme.hu

Optional Items

Bug tracker:
Home page:
Repository:
Tags:

Flag the plugin as experimental

Súgó Mégsem OK

2. ábra Plugin Builder modul párbeszédablaka

Plugin Builder Results


Your plugin **RealCentroid** was created in:
/home/siki/realcentroid/RealCentroid

Your QGIS plugin directory is located at:
/home/siki/.qgis2/python/plugins

What's Next

1. Copy the entire directory containing your new plugin to the QGIS plugin directory
2. Compile the ui file using pyuic4
3. Compile the resources file using pyrcc4
4. Test the plugin by enabling it in the QGIS plugin manager
5. Customize it by editing the implementation file **realcentroid.py**
6. Create your own custom icon, replacing the default **icon.png**
7. Modify your user interface by opening **realcentroid.ui** in Qt Designer (don't forget to compile it with pyuic4 after changing it)
8. You can use the **Makefile** to compile your Ui and resource files when you make changes. This requires GNU make (gmake)

For more information, see the PyQGIS Developer Cookbook at: <http://www.qgis.org/pyqgis-cookbook/index.html>.

 ©2013 GeoApt LLC - geoapt.com

OK

3. ábra Plugin Builder listája a további teendőkről

Mozgassuk át az új modult a `.qgis2/python/plugins` könyvtárba (az egyes könyvtárak nevét a fenti ablak tartalmazza):

```
mv /home/siki/realcentroid/RealCentroid /home/.qgis2/python/plugins
```

A *Plugin Builder* modul számos fájlt és könyvtárat hozott létre:

help	Makefile	README.txt	ui_realcentroid.ui
i18n	metadata.txt	realcentroiddialog.py	
icon.png	plugin_upload.py	realcentroid.py	
__init__.py	README.html	resources.qrc	

A `help` könyvtárba a modulunk sűgője kerülhet, ilyen most nem készítünk. Az `i18n` könyvtárba a különböző nyelvekre lefordított üzenetek kerülhetnek, most ilyen sem készítünk. Az `icon.png` a modulunk ikonját tartalmazza, ezt szabadon módosíthatja egy képszerkesztő programmal (pl. *Gimp*), a méretét ne módosítsa.

Az `__init__.py` fájl tartalmazza a modul inicializálásához szükséges Python kódot, ezt nem kell módosítanunk. Ugyanezeket az adatokat tartalmazza a `metadata.txt` fájl. Az `__init__.py` fájlt a korábbi QGIS változatok használják.

A `plugin_upload.py` szkript a modul feltöltését teszi lehetővé a QGIS modulok közé.

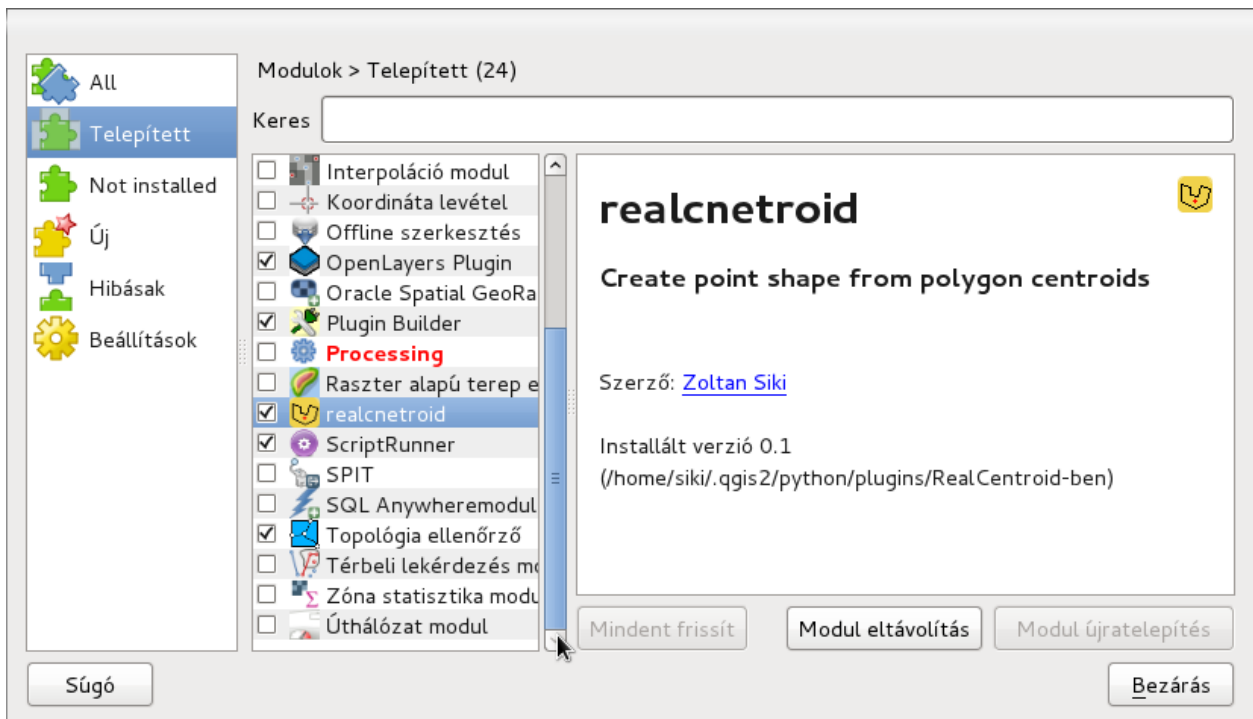
A `README.html` és `README.txt` fájlok a modulra vonatkozó a felhasználók számára fontos információkat tartalmazhatnak. A *Plugin Builder* futtatása után a modullal kapcsolatos további teendőket tartalmazza. Tartalmuk szabadon módosítható.

A `realcentroiddialog.py` fájl a párbeszédablak kezelésével kapcsolatos Python kódot tartalmazza, a `realcentroid.py` fájl pedig modulunk kódját (mindkét fájl egy-egy új osztályt hoz létre, melyet a későbbiek során bővíteni fogunk).

A `Makefile` a modult futtatásra elkészítő szkript. Rögtön futtassuk le a `make` parancs segítségével, előtte lépjen be a modul könyvtárba. Ez a következő két parancsot fogja végrehajtani:

```
pyuic4 -o ui_realcentroid.py ui_realcentroid.ui
pyrcc4 -o resources_rc.py resources.qrc
```

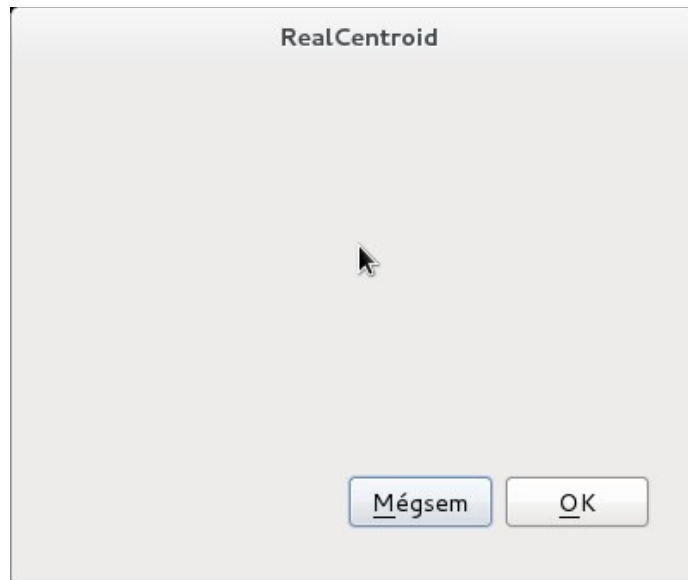
A `pyuic4` parancs a Qt ui fájlt Python kóddá alakítja át. Az `ui_realcentroid.ui` fájl a modul párbeszédablakának definícióját tartalmazza. A `pyrcc4` a modulhoz tartozó erőforrásokat alakítja át Python kóddá, az erőforrásokat a `resources.qrc` fájl tartalmazza, esetünkbe csak a modul ikonját adja meg. Mindkét forrás fájl (`ui_realcentroid.ui` és `resources.qrc`) XML kódot tartalmaz, melyeket Python fájlkká alakítja át a `make` parancs. A `make` parancs kiadását meg kell ismételniünk amikor ennek a két fájlnak a tartalma módosul.



4. ábra Az új modul bekapcsolása

Most már ki is próbálhatjuk a modulunkat. Indítsa el a QGIS-t, a *Modulok/Modul kezelés és telepítés* menüben már megjelenik a modulunk (én titokban módosítottam az ikont).

Kapcsolja be a modult, keresse meg a modul ikonját vagy a *Modulok* menüből válassza ki a *Real centroids* menüpontot. Egy párbeszédablak jelenik meg a képernyőn, mely két gombot (OK és Mégsem) tartalmaz.



5. ábra Az új modul párbeszédablaka

Eddig a pontig minden modul létrehozása azonos. A következőkben a speciális feladat megoldására koncentrálnunk.

A modul kibővítése

A további konkrét programozás előtt tervezzük meg a modul működését. A centrálisok generálásához először ki kell választani a betöltött felület típusú rétegek közül egyet és a készítendő pont típusú shape fájl nevét és könyvtárát kell megadni. Ezt a párbeszédablak bővítésével tesszük meg.

A párbeszédablak kitöltése után a modulunknak először létre kell hoznia az új pont shape fájlt, majd a kiválasztott felület típusú réteg elemein egyesével végig kell menni és a centrális koordinátáit meg kell határozni.

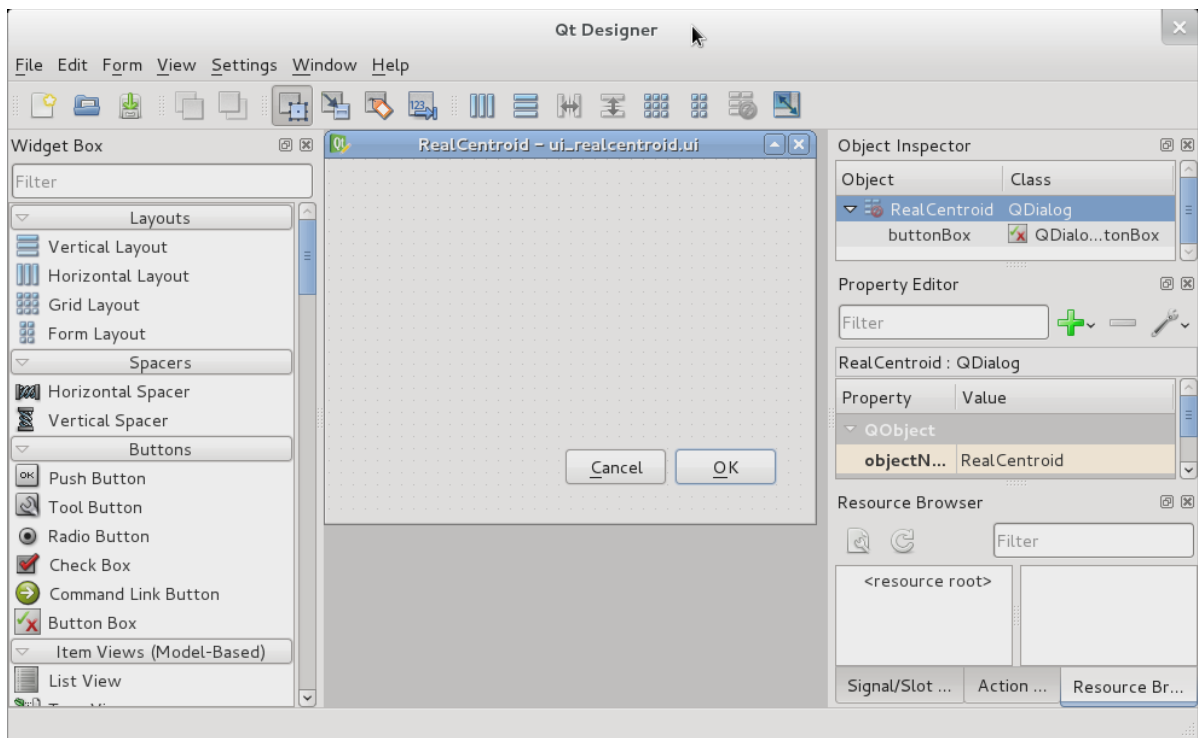
A centrális generáláshoz, mint ahogy a bevezetőben írtam, nincs kész metódusa a QGIS-nek. A következő stratégiát fogjuk követni:

- az elem súlypontját kiszámoltatjuk a QGIS beépített metódusával
- ha a súlypont nem esik bele a felületbe, akkor felület három szomszédos pontjának súlypontjával próbálkozunk, amíg egy belső pontot találunk
- a belső pontot és a felület elem attribútumait kiírjuk a pont shape fájlba.

A párbeszédablak elkészítése

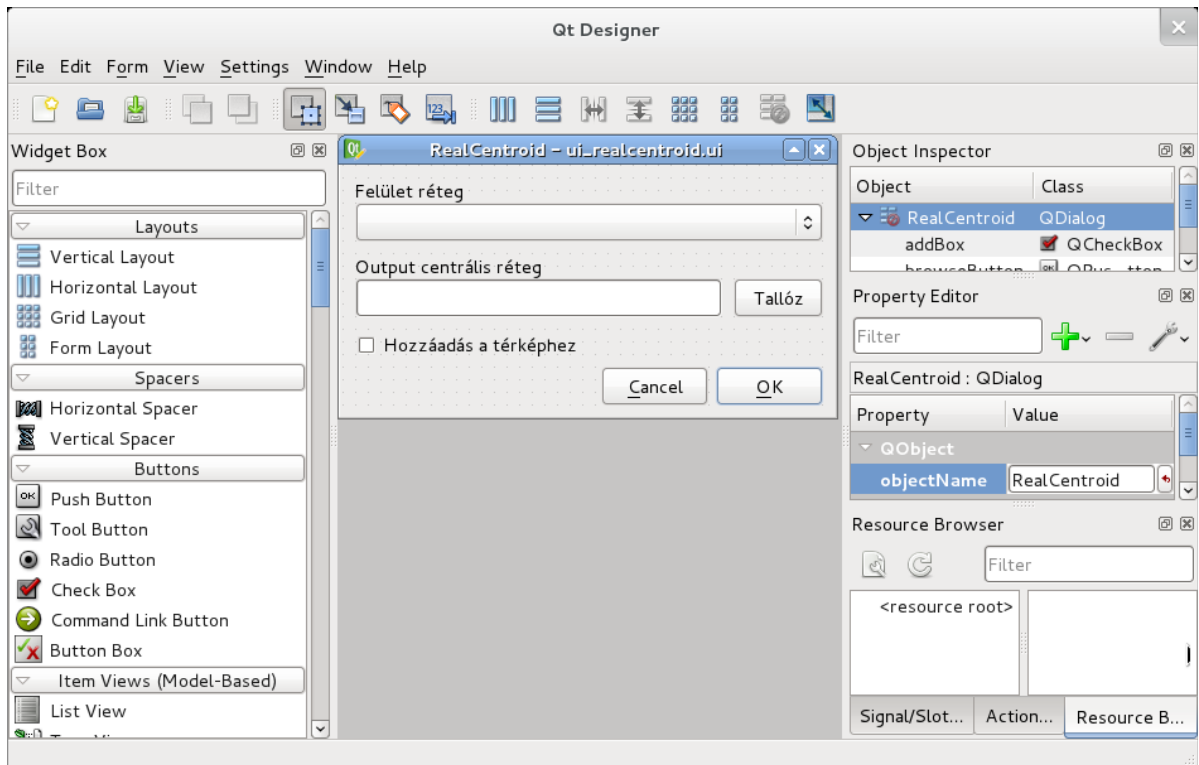
A párbeszédablak szerkesztéséhez a **Qt4 Designer** programot használjuk. Ezt előzőleg telepíteni kell a gépünkre. A **Qt4 Designer** elindítása után nyissuk meg a modulunkhoz tartozó *ui_realcentroid.ui* fájlt.

Először adjunk egy címkét és egy legördülő listát a párbeszédablakunkhoz. A bal oldali Widget Box-ban keressük meg a *Label* elemet és húzza be az párbeszédablakunkba. A címkére duplán kattintva írjuk át a szöveget „Felület réteg”-re, módosítsuk a címke szélességét, hogy elférjen a szöveg. Hasonló módon adjunk egy *Combo Box* elemet az ablakhoz. Az *objectName* tulajdonságát a jobb oldali tulajdonságok között írjuk át *layerBox*-ra (ezzel a névvel hivatkozhatunk az elemre a programunkból). Növeljük meg ennek az elemnek is a szélességét.



6. ábra A párbeszédablak a Qt4 Designerben

Folytassuk az ablak feltöltését egy második címkével és egy *Line Edit* elemmel. A *Line Edit* elem név tulajdonságát módosítsuk *pointEdit*-re. Tegyük egy nyomógombot (Push Button) a szerkesztő sor mellé, a feliratát módosítsuk „Tallóz”-ra, neve legyen *browseButton*. Végül adjunk egy jelölő négyzetet (Check box) az ablakhoz, a neve legyen *addBox*, a felirata pedig „Hozzáadás a térképhez”. Ha a felhasználó bekapcsolja a jelölő négyzetet, akkor a centrális réteget a programunk hozzáadja az aktuális térképhez. A párbeszédablak elemeit rendezze el az alábbi ábrának megfelelően. A párbeszédablak tulajdonságai között kapcsoljuk be a *modal* tulajdonságot. (ez azt jelenti, hogy amíg a párbeszédablakot nem zárja le, a QGIS nem reagál az ezen kívüli felhasználói műveletekre).



7. ábra A kész párbeszédablak

Futtassuk újra a *make* parancsot a modul könyvtárában, hogy a módosított párbeszédablak Python kódját

elkészítsük, majd nézzük meg a modulunkat a QGIS-ben. A szöveg mezőt már szerkesztheti, a jelölő négyzetet be- és kikapcsolhatja, a réteg lista üres, a Tallóz gomb megnyomására nem történik semmi. A réteg lista feltöltéséhez és a tallózáshoz a programot kell bővítenünk.

Megjegyzés

A továbbiakban, ha nem lépünk ki a QGIS-ből, akkor a modul kódjában történt módosításokat nem veszi észre a program. Ehhez a modul kezelővel a modul ki kell kapcsolni, majd a modul kezelőbe újra belépve be kell kapcsolni. Használhatja a *Plugin reloader* modult is, akkor egy kattintással újra beolvashatja a modult. Egy kicsit lassabb megoldás, ha kilépünk a QGIS-ből és újraindítjuk.

Először készítsük el azt a kódrészletet, mely adott típusú rétegek listáját adja vissza. Ez egy új Python fájlba írjuk be *util.py*, mivel ez más modulban is hasznos lehet. Nyissuk meg a kedvenc szövegszerkesztőnkkel (pl. Vim, Notepad++, UltraEdit). Importáljunk két modult, melyre szükségünk lesz, majd készítsük el azt a függvényt, mely a paraméterlistáján egy listában megadott típusú rétegek nevét adja vissza az aktuális QGIS projektből.

```
from qgis.core import *
import locale

# Return list of names of layers in QgsMapLayerRegistry
# vTypes - list of layer types allowed
#         (e.g. Qgs.Point, Qgs.Line, Qgs.Polygon or "all" or "raster")
# return sorted list of layer names
def getLayerNames(vTypes):
    layermap = QgsMapLayerRegistry.instance().mapLayers()
    layerlist = []
    if vTypes == "all":
        for name, layer in layermap.iteritems():
            layerlist.append(unicode(layer.name()))
    else:
        for name, layer in layermap.iteritems():
            if layer.type() == QgsMapLayer.VectorLayer:
                if layer.geometryType() in vTypes:
                    layerlist.append(unicode(layer.name()))
            elif layer.type() == QgsMapLayer.RasterLayer:
                if "raster" in vTypes:
                    layerlist.append(unicode(layer.name()))
    return sorted(layerlist, cmp=locale.strcoll)
```

`QgsMapLayerRegistry.instance().mapLayers()` visszaadja a projekt rétegeinek nevét. A további kód ezek közül kiválasztja a kívánt típusúakat. Adjuk hozzá az új forrás fájlt a *Makefile*-hoz, *SOURCES* sort bővítsük:

```
SOURCES = realcentroid.py ui_realcentroid.py __init__.py realcentroiddialog.py util.py
```

Most építsük be a rétegek listáját visszaadó függvényünket a párbeszédablakunkba, hívjuk meg a megjelenítés előtt és töltsük fel a legördülő listánkat. A párbeszédablak objektumunk kódját a *realcentroiddialog.py* fájl tartalmazza. Nyissuk meg egy szövegszerkesztővel, és bővítsük az alábbiak szerint.

```
from PyQt4 import QtCore, QtGui
from ui_realcentroid import Ui_RealCentroid
from qgis.core import *
import util

# create the dialog for real centroids
class RealCentroidDialog(QtGui.QDialog):
    def __init__(self):
        QtGui.QDialog.__init__(self)
        # Set up the user interface from Designer.
        self.ui = Ui_RealCentroid()
        self.ui.setupUi(self)
```



```
def showEvent(self, event):
    # remove previous entries from layer list
    self.ui.layerBox.clear ()
    # add polygon layers to list
    self.ui.layerBox.addItem(util.getLayerNames([Qgis.Polygon]))
```

A kódlistánban elszűrítettük azokat a sorokat, melyeket a *Plugin Builder* már korábban beletett. A `showEvent` függvényt automatikusan meghívja a párbeszédablak a megjelenítése előtt. Először töröljük a legördülő réteglista tartalmát, majd az aktuális réteglistával bővítjük.

A következő lépésben készítsük el a *Tallóz* gombhoz tartozó kódot. A gomb megnyomásakor egy fájl kiválasztó ablakot jelenítünk meg. Ehhez először kapcsoljunk egy függvényt a gomb kattintás eseményéhez:

```
QObject.connect(self.ui.browseButton, SIGNAL("clicked()"), self.browse)
```

A fenti sort az `__init__` metódus végére tegyük be. Ennek hatására, ha a *Tallóz* gombra kattintunk, akkor az objektum `browse` függvénye automatikusan végrehajtódik. A `browse` függvény a következő:

```
def browse(self):
    self.ui.pointEdit.clear() # output szöveg mező törlése
    # fájl kiválasztó párbeszédablak megnyitása
    (self.shapefileName, self.encoding) = util.saveDialog(self)
    if self.shapefileName is None or self.encoding is None:
        return
    self.ui.pointEdit.setText(self.shapefileName) # output szövegmező feltöltése
```

A `browse` függvényt akkor kell meghívni, amikor a *Tallóz* gombra kattint a felhasználó. Ezt úgy érjük el, hogy a gomb kattintás esemény kezelésére beállítjuk a függvényünket. Ezt a párbeszédablak `__init__` függvényében célszerű megtenni, a `QtCore.QObject.connect` függvény segítségével.

```
def __init__(self):
    QtGui.QDialog.__init__(self)
    # Set up the user interface from Designer.
    self.ui = Ui_RealCentroid()
    self.ui.setupUi(self)
    # add action to browse button
    QtCore.QObject.connect(self.ui.browseButton, QtCore.SIGNAL("clicked()"),
self.browse)
```

A `browse` metódust még nem tudjuk kipróbálni, mert meghívja a `saveDialog` függvényt, amit még nem készítettünk el. Ezt a `util.py` fájlba helyezzük el, mert esetleg más programunkban is fel tudjuk használni majd. Egy újabb modulból, a `qgis.gui` használunk egy objektum osztályt, amit jeleznünk kell a fájl elején.

```
from PyQt4.QtCore import *
from PyQt4.QtGui import *
from qgis.core import *
from qgis.gui import QgsEncodingFileDialog
import locale

...

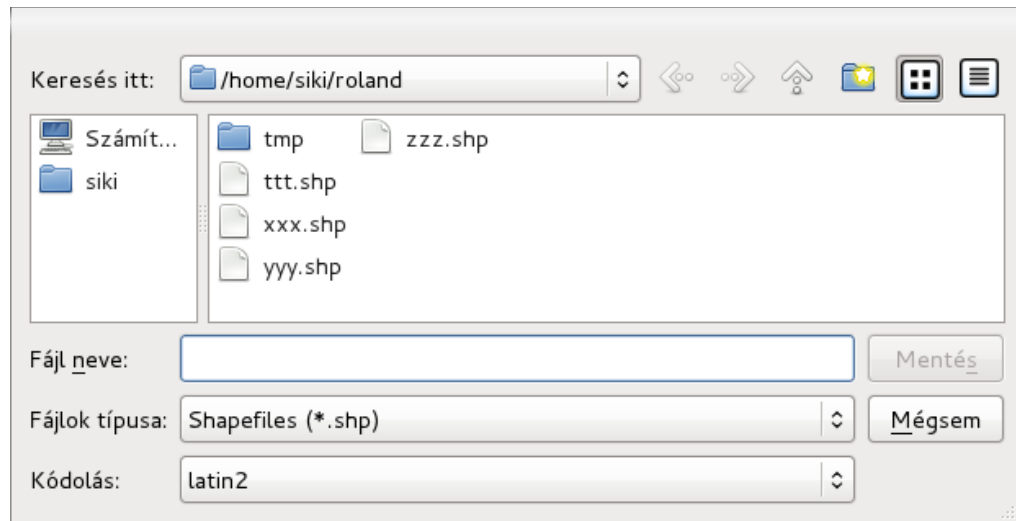
# Fájl mentés dialóg a karakter kódolás kiválasztásával
def saveDialog(parent, filtering="Shapefiles (*.shp)":
    # előző beállítások beszerzése
    settings = QSettings()
    dirName = settings.value("/UI/lastShapefileDir")
    encode = settings.value("/UI/encoding")
    # dialóg létrehozása
    fileDialog = QgsEncodingFileDialog(parent, "Output shape file", dirName, filtering,
encode)
    fileDialog.setDefaultSuffix("shp") # alapértelmezett kiterjesztés .shp
    fileDialog.setFileMode(QFileDialog.AnyFile) # nem létező fájl is megadható
    fileDialog.setAcceptMode(QFileDialog.AcceptSave) # fájl mentés mód
    fileDialog.setConfirmOverwrite(True) # felülírás megerősítést kér
```

```

if not fileDialog.exec_() == QDialog.Accepted:
    return None, None # mégsem gombbal léptek ki
files = fileDialog.selectedFiles() # kiválasztott fájlnevek listája
settings.setValue("/UI/lastShapefileDir",
QFileInfo(unicode(files[0])).absolutePath()) # beállítások megőrzése
# első fájlnev és karakterkódolás visszaadása
return (unicode(files[0]), unicode(fileDialog.encoding()))

```

Most már kipróbálhatjuk a *Tallóz* gombot a programunkban. A gomb megnyomása után egy új párbeszédablak jelenik meg, melyben megadhatjuk a fájl nevét és kódolását. Az ablak kinézete az alkalmazott operációs rendszer függvényében eltérő lehet.



8. ábra Fájl megadás ablak

Mielőtt a modulunk lényegi részét elkezdenénk megírni, még egy egyszerű függvényre van szükségünk, mely a réteg neve alapján visszaadja az ahhoz kapcsolódó objektumot. Az alábbi függvényt szintén a *util.py* fájlba helyezzük el. Rögtön próbáljuk ki!

```

# Réteg azonosító beszerzése a réteg név alapján
def getMapLayerByName(myName):
    layermap = QgsMapLayerRegistry.instance().mapLayers() # minden réteg
    for name, layer in layermap.iteritems(): # egyesével végigmegegyünk a rétegeken
        if layer.name() == myName:
            if layer.isValid():
                return layer
            else:
                return None
    return None

```

Akkor most hozzáláthatunk az elején megfogalmazott cél megvalósításához. Tervezzük meg az algoritmusunkat. A kiválasztott felület típusú réteg elemein egyesével végig kell mennünk, egy pont elemet létre kell hoznunk, mely a felületen belül esik, majd végül ki kell írni a pontot a felülethez tartozó attribútumokkal együtt az új shape fájlba. Hogyan tudunk a felületbe eső pontot generálni? A QGIS API-ban a felület objektumhoz létezik egy centroid metódus. Írjuk meg ennek felhasználásával az új shape fájl létrehozását.

```

# create centroids shape file
def centroids(self):
    # réteg objektum beszerzése
    vlayer = util.getMapLayerByName(self.dlg.ui.layerBox.currentText())
    # a réteg adatainak eléréséhez szükséges objektum
    vprovider = vlayer.dataProvider()
    # az új shape fájlba íráshoz szükséges objektum létrehozása,
    # az attribútumok és a vetület megegyezik a felület réteggel
    writer = QgsVectorFileWriter(self.dlg.shapefileName, self.dlg.encoding,
vprovider.fields(), QGis.WKBPoint, vprovider.crs())

```



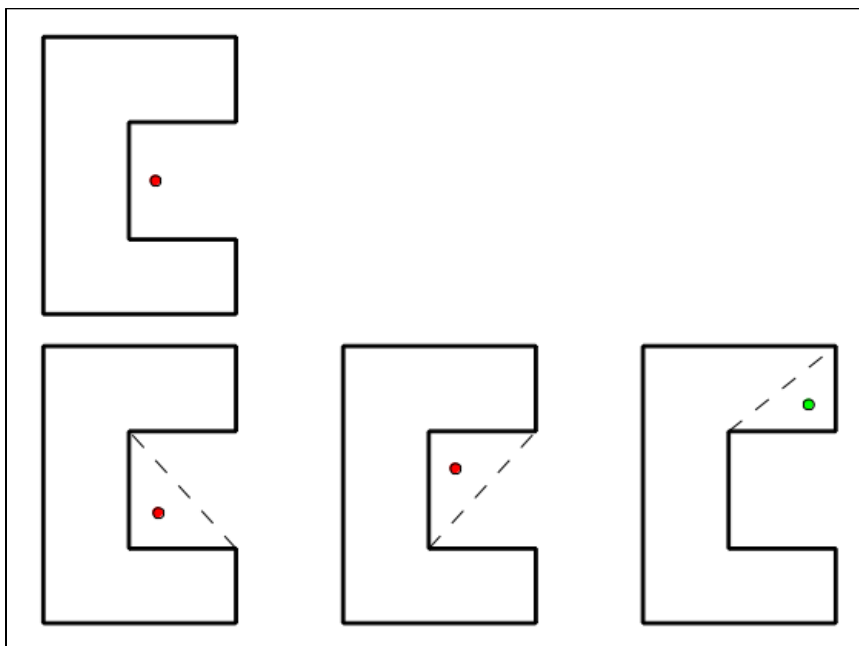
```

# az aktuális felület elem tárolásához szükséges objektum
inFeat = QgsFeature()
# az aktuális pont elem (centrális) tárolásához szükséges objektum
outFeat = QgsFeature()
# összes felület elemhez iterátor létrehozása
fit = vprovider.getFeatures()
# a felület elemek feldolgozása egyesével
while fit.nextFeature(inFeat):
    # aktuális felületelem geometriájának beszerzése
    inGeom = inFeat.geometry()
    # aktuális felület elem attribútumai
    atMap = inFeat.attributes()
    # pont geometria létrehozása
    outGeom = inGeom.centroid()
    # új pont elem attribútumainak és geometriájának beállítása
    outFeat.setAttributes(atMap)
    outFeat.setGeometry(outGeom)
    # centrális kiírása shape fájlba
    writer.addFeature(outFeat)
del writer

```

A fenti kódrészlet a centroid függvény viselkedése miatt, nem működik tökéletesen azokban az esetekben amikor a súlypontot nem tartalmazza a felület. Ugyanez a megoldás található meg az *fTools* modulban (*Vektor/Geometria eszközök/Felület centrálisok* menüpont).

A számos lehetséges megoldás közül azt az egyszerű megoldást választjuk, hogy a felület határán lévő három szomszédos pont centrálisát választjuk, ha beleesik az eredeti felületbe különben egy ponttal továbblépünk a határpontokon.



9. ábra A belső pont keresés algoritmus

```

outGeom = inGeom.centroid()
# a súlypont kívül esik?
if not inGeom.contains(outGeom):
    # pontok listájának beszerzése
    if inGeom.wkbType() == Qgs.WKBPolygon:
        poly = inGeom.asPolygon()
    elif inGeom.wkbType() == Qgs.WKBMultiPolygon:
        poly = inGeom.asMultiPolygon()
    # az egyes külső vagy belső határokra
    for part in poly:
        x1 = part[0][0]

```

```

        y1 = part[0][1]
        x2 = part[1][0]
        y2 = part[1][1]
        for i in range(2, len(part)):
            x3 = part[i][0]
            y3 = part[i][1]
            # felület három szomszédos pontra
            tmpGeom = QgsGeometry.fromPolygon([[QgsPoint(x1, y1),
QgsPoint(x2, y2), QgsPoint(x3, y3)]])
            outGeom = tmpGeom.centroid()
            found = inGeom.contains(outGeom)
            if found:
                break
            x1, y1 = x2, y2
            x2, y2 = x3, y3
            # kilépés a ciklusból
            # pontok továbbléptetése
        if found:
            break
    outFeat.setAttributes(atMap)

```

Most más „csak” az eredmény centrális shape fájl betöltéséről kell gondoskodnunk, ha a felhasználó bejelölte a *Hozzáadás a térképhez* négyzetet. Ehhez először készítsünk egy általános OGR réteg térképhez adás függvényt a *util.py* fájlban.

```

# shape hozzáadás a térképhez
def addShape(shapefile_path):
    file_info = QFileInfo(shapefile_path)
    # létezik a fájl?
    if file_info.exists():
        layer_name = file_info.completeBaseName() # név könyvtár és kiterj. nélkül
    else:
        return False
    vlayer_new = QgsVectorLayer(shapefile_path, layer_name, "ogr") # réteg létrehozása
    if vlayer_new.isValid():
        QgsMapLayerRegistry.instance().addMapLayer(vlayer_new) # réteg a térképre
        return True
    else:
        return False

```

A fenti függvény igaz értéket (True) ad vissza, ha sikerült a réteg betöltése. Most már csak a *centroids* metódust kell kiegészítenünk a *realcentroid.py* fájlban. Adjuk a következő sorokat a *centroids* metódus végére.

```

    del writer
    # add centroid shape to canvas
    if self.dlg.ui.addBox.checkState() == Qt.Checked:
        if not util.addShape(self.dlg.shapefileName):
            QMessageBox.warning(self, "RealCentroid", "Error loading shapefile:\n" +
self.dlg.shapefileName)

```

Most már csak a *run* metódust kell kiegészítenünk, hogy az **OK** gomb megnyomása után fusson le a *centroids*.

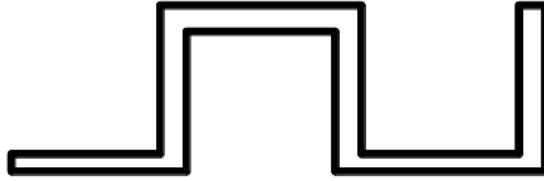
```

# run method that performs all the real work
def run(self):
    # show the dialog
    self.dlg.show()
    # Run the dialog event loop
    result = self.dlg.exec_()
    # See if OK was pressed
    if result == 1:
        self.centroids()

```

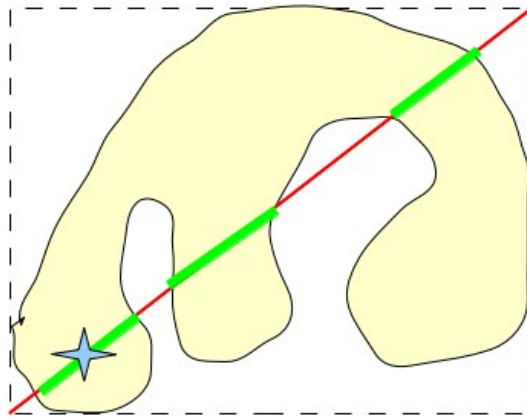
Ezzel elkészült a modulunk 0.1 verziója. Mi legyen a folytatás?

1. Teszteljük a modult mielőtt továbbadnánk a felhasználónak! Működik az algoritmusunk az alábbi felület esetén?



10. ábra Működik erre a felületre az algoritmusunk?

2. Javítsuk a modul hibakezelését! Mi történik, ha nem sikerül létrehozni az eredmény centrális shape fájlt jogosultsági probléma miatt?
3. Készítsünk súgót a modulhoz!
4. Tegyük nemzetközivé a modulunkat (az üzenetek több nyelven is megjelenhessenek)!
5. Vizsgáljuk a modul eredményét és hatékonyságát, próbáljunk javítani rajta. Például hogyan javíthatnánk azon, hogy a generált centrálisok ne legyenek túl közel a felület határához? Nem lenne jobb, ha a felület és a befoglaló négyzet átlójának a metszetét határozzuk meg, majd a centrális az eredmény vonalszakasz felezőpontja legyen?



11. ábra Módosított algoritmus

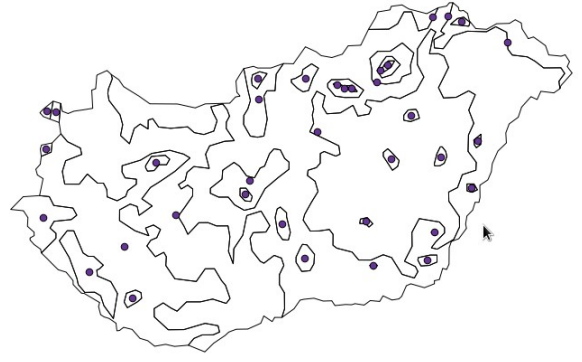
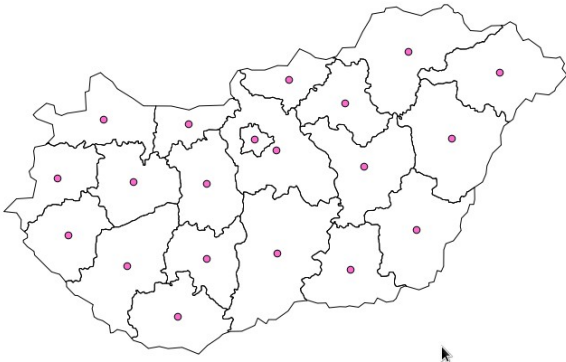
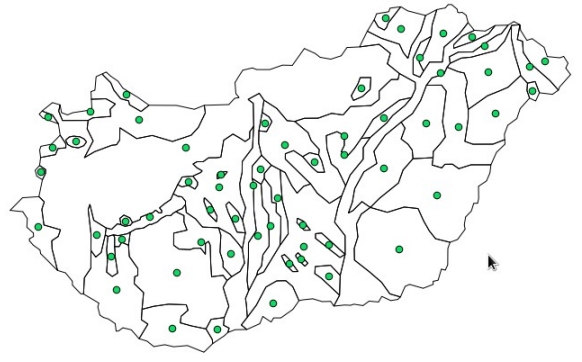
6. Reagáljunk a felhasználói észrevételekre!
7. Nézegezzünk más által írt modulokat és tanuljunk belőle!

Hasznos linkek:

http://www.qgis.org/en/docs/pyqgis_developer_cookbook/index.html

<http://www.qgis.org/api/>

A modul teljes forráskódja letölthető a <http://www.agt.bme.hu/qgis/qgis/realcentroid.zip> címről. Tömörítse ki a bejelentkezési könyvtárának `.qgis2/python/plugin/RealCentroid` könyvtárába.



12. ábra Tesztelési eredmények