GIS and Irradiation Mapping

Francesco Anselmo
francesco.anselmo@arup.com
vector data / points, lines, polygons, shapes
raster data / LiDAR data
Arup Irradiation Mapping Tool (iMAP)

- Integration of Geographic Information Systems (GIS) and the Radiance enables automation and highly efficient analysis of individual buildings as well as large areas with tens of thousands of addresses.
Why Shapefiles?

A shapefile stores [nontopological geometry and attribute information](#) for the spatial features in a data set. The geometry for a feature is stored as a shape comprising a set of vector coordinates.

Because shapefiles do not have the processing overhead of a topological data structure, they have advantages over other data sources such as faster drawing speed and edit ability. Shapefiles handle single features that overlap or that are noncontiguous. They also typically require less disk space and are easier to read and write.

Shapefiles can support point, line, and area features. Area features are represented as closed loop, double-digitized polygons. Attributes are held in a dBASE® format file. Each attribute record has a one-to-one relationship with the associated shape record.

---

**ESRI Shapefile Technical Description**

*An ESRI White Paper—July 1998*

**Table 1**

Description of the Main File Header

<table>
<thead>
<tr>
<th>Position</th>
<th>Field</th>
<th>Value</th>
<th>Type</th>
<th>Byte Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>File Code</td>
<td>9994</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 4</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 8</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 12</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 16</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 20</td>
<td>Unused</td>
<td>0</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 24</td>
<td>File Length</td>
<td>File Length</td>
<td>Integer</td>
<td>Big</td>
</tr>
<tr>
<td>Byte 28</td>
<td>Version</td>
<td>1000</td>
<td>Integer</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 32</td>
<td>Shape Type</td>
<td>Shape Type</td>
<td>Integer</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 36</td>
<td>Bounding Box Xmin</td>
<td>Xmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 44</td>
<td>Bounding Box Ymin</td>
<td>Ymin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 52</td>
<td>Bounding Box Xmax</td>
<td>Xmax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 60</td>
<td>Bounding Box Ymax</td>
<td>Ymax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 68*</td>
<td>Bounding Box Zmin</td>
<td>Zmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 76*</td>
<td>Bounding Box Zmax</td>
<td>Zmax</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 84*</td>
<td>Bounding Box Mmin</td>
<td>Mmin</td>
<td>Double</td>
<td>Little</td>
</tr>
<tr>
<td>Byte 92*</td>
<td>Bounding Box Mmax</td>
<td>Mmax</td>
<td>Double</td>
<td>Little</td>
</tr>
</tbody>
</table>

* Unused, with value 0.0, if not Measured or Z type

---

**Shape Type**

- 0: Null Shape
- 1: Point
- 3: PolyLine
- 5: Polygon
- 8: MultiPoint
- 11: PointZ
- 13: PolyLineZ
- 15: PolygonZ
- 18: MultiPointZ
- 21: PointM
- 23: PolyLineM
- 25: PolygonM
- 28: MultiPointM
- 31: MultiPatch
• Commercial
  • ArcGIS

• Open Source
  • QGIS
  • Grass
  • GDAL-OSR
  • pyshp
  • shapely
  • scipy/numpy/matplotlib
  • libLAS
  • mapnik
Overview

NOTE: This project has been migrated to Github. This google code site will be available indefinitely. As of June 16, 2014, the source and some wiki pages have been migrated over.

This library reads and writes ESRI Shapefiles in pure Python. You can read and write shp, shx, and dbf files with all types of geometry. Everything in the public ESRI shapefile specification is implemented. This library is compatible with Python versions 2.4 to 3.x.

Get Started Instantly

1. Download shapefile.py
2. Start Python
3. import shapefile
4. Try one of the examples below

OR

Just run: easy_install pyshp

OR

pip install pyshp

If you are looking for information on .sbn and .sbx file formats some documentation is available here.

Latest News

8/8/2013 - Please upgrade to PyShp 1.1.9 which fixes issues with polylines, polygons, as well as some z-value corner cases. This update resolves (issue-54), (issue-56), and (issue-66).

6/23/2013 - Released PyShp 1.1.7! This release fixes several bugs including (issue-49), (issue-37), (issue-26), and (issue-22). Other improvements include:

- Added Python geo_interface convention to export shapefiles as GeoJSON.
- Used is_string() method to detect file names passed as unicode strings.
- Added Reader.iterShapes() method to iterate through geometry records for parsing large files efficiently.
- Added Reader.iterRecords() method to iterate through dbf records efficiently in large files.
- Modified shape() method to use iterShapes() if shx file is not available.
- Fixed bug which prevents writing the number 0 to dbf fields.
- Updated shape() method to calculate and seek the start of the next record. The shapefile spec does not require the content of a geometry record to be as long as the content length defined in the header. The result is you can delete features without modifying the record header allowing for empty space in records.
Reading Points in Shapes

>>> import shapefile
>>> sf = shapefile.Reader("shapefiles/blockgroups")
>>> shapes = sf.shapes()
>>> # Read the bounding box from the 4th shape
>>> shapes[3].bbox
[-122.485792, 37.786031000000003, -122.446285, 37.811019000000002]

>>> # Read the 8th point in the 4th shape
>>> shapes[3].points[7]
[-122.471063, 37.78402999999998]

Reading Database Attributes

>>> # Read the field descriptors for the database file
>>> sf.fields
[['"DeletionFlag", "C", 1, 0], ["AREA", "N", 18, 5],
... ["BKG_KEY", "C", 12, 0], ["POP1990", "N", 9, 0], ["POP90_SQMI", "N", 10, 1],
... ["HOUSEHOLD", "N", 9, 0],
... ["MALES", "N", 9, 0], ["FEMALES", "N", 9, 0]]

>>> # Read the 2nd and 3rd field values of the 4th database record
>>> sf.records[3][1:3]
['060750601001', 4715]

Writing Shapefiles

>>> import shapefile

>>> # Make a point shapefile
>>> w = shapefile.Writer(shapefile.POINT)
>>> w.point(90.3, 30)
>>> w.point(92, 40)
>>> w.point(-122.4, 30)
>>> w.point(-90.3, 35.1)
>>> w.field('FIRST_FLD','C','40')
>>> w.field('SECOND_FLD','C','40')
>>> w.records(['First','Point'])
>>> w.records(['Second','Point'])
>>> w.records(['Third','Point'])
>>> w.records(['Fourth','Point'])
>>> w.save('shapefiles/test/point')

>>> # Create a polygon shapefile
>>> w = shapefile.Writer(shapefile.POLYGON)
>>> w.poly(parts=[[1,1],[5,5],[5,1],[3,3],[1,1]])
>>> w.field('FIRST_FLD','C','40')
>>> w.field('SECOND_FLD','C','40')
>>> w.record(['First','Polygon'])
>>> w.save('shapefiles/test/polygon')
PROJCS["WGS_1984_UTM_Zone_30N",
    GEOGCS["GCS_WGS_1984",
        DATUM["D_WGS_1984",
            SPHEROID["WGS_1984",6378137.0,298.257223563]],
            PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["False_Easting",500000.0],
    PARAMETER["False_Northing",0.0],
    PARAMETER["Central_Meridian",-3.0],
    PARAMETER["Scale_Factor",0.9996],
    PARAMETER["Latitude_Of_Origin",0.0],
    UNIT["Meter",1.0]]
from sys import argv, exit, stdout
from os.path import exists, isdir, join, basename, splitext
import shapefile
from os import system, mkdir, listdir
from commands import getoutput
from subprocess import Popen
from optparse import OptionParser
from shapely.geometry import Polygon
from math import sqrt, atan, asin, acos, degrees, radians, pi
from scipy import matrix
from numpy import dot, cross
import vtk
import pyproj
from osgeo import ogr
import mapnik
import PIL import Image
sf = get_options() #, material
shapefilename, shapefileext = get_name_ext(sf)
print shapefilename, shapefileext
if exists(sf):
    # get barycentre from bounding box
    rectangle = getbbox(shapefilename)
    print '%s bounding box:' % sf, rectangle
    orig_X = (rectangle[2]+rectangle[0])/2
    orig_Y = (rectangle[3]+rectangle[1])/2
    driver = ogr.GetDriverByName('ESRI Shapefile')
    # get source projection
    ds = driver.Open(sf)
    layer = ds.GetLayer()
    sr = layer.GetSpatialRef()
    print 'source projection:', sr.ExportToProj4()
    print 'UTM Zone:', sr.GetUTMZone()
    srcProj = pyproj.Proj(sr.ExportToProj4())
    dstProj = pyproj.Proj(proj='longlat', ellps='WGS84', datum='WGS84')
    print 'dest projection:', dstProj.srs
    longitude, latitude = pyproj.transform(srcProj, dstProj, orig_X, orig_Y)
    dest_X, dest_Y = pyproj.transform(dstProj, srcProj, out_lon, out_lat)
    dest_long, dest_lat = pyproj.transform(srcProj, dstProj, out_x, out_y)
    print "UTM zone %s coordinate x, y (%0.4f, %0.4f) = (%0.4f, %0.4f) lat, long " % (sr.GetUTMZone(), orig_X, orig_Y, latitude, longitude)
    print "lat, long coordinate x, y (%0.4f, %0.4f) = (%0.4f, %0.4f) lat, long " % (out_lat, out_lon, dest_X, dest_Y)
    print "UTM zone %s coordinate x, y (%0.4f, %0.4f) = (%0.4f, %0.4f) lat, long " % (sr.GetUTMZone(), out_x, out_y, dest_lat, dest_long4)
Automatic generation of 3D model and sampling points on roofs from GIS shape-files
Terrain and obstructions model from LIDAR
Combined geometry and terrain model for more accurate analysis
Local and background obstructions / horizon profiles
Irradiation components: direct and diffuse
Global annual irradiation
Instantaneous irradiance profiles
UK roofs
Irradiance data into GIS
Irradiance data into GIS
### Interrogation of GIS model

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>irradiation</td>
<td>GBRBOUR0001438</td>
</tr>
<tr>
<td>AREA2D_m2</td>
<td>80.7150885018</td>
</tr>
<tr>
<td>AREA3D_m2</td>
<td>109.081052522</td>
</tr>
<tr>
<td>AZIMUTH</td>
<td>255.762538049</td>
</tr>
<tr>
<td>Area</td>
<td>80.7150885007</td>
</tr>
<tr>
<td>BLOCKID</td>
<td>3</td>
</tr>
<tr>
<td>BUILDINGID</td>
<td>GBRBOUR0001438</td>
</tr>
<tr>
<td>CO2_REDUCTION</td>
<td>8.31714699412</td>
</tr>
<tr>
<td>ENERGY</td>
<td>15283.816006</td>
</tr>
<tr>
<td>FLATROOF</td>
<td>0</td>
</tr>
<tr>
<td>INCLINATION</td>
<td>42.2724035898</td>
</tr>
<tr>
<td>IRR_GLOBAL</td>
<td>1030.25225229</td>
</tr>
<tr>
<td>IRR_SKY</td>
<td>485.273070642</td>
</tr>
<tr>
<td>IRR_SUN</td>
<td>544.979181651</td>
</tr>
<tr>
<td>NX</td>
<td>-0.16543386475</td>
</tr>
<tr>
<td>NY</td>
<td>-0.651995391849</td>
</tr>
<tr>
<td>NZ</td>
<td>0.739955164454</td>
</tr>
<tr>
<td>REPAIR</td>
<td>0</td>
</tr>
<tr>
<td>ROOFSUB</td>
<td>1</td>
</tr>
<tr>
<td>ROOFTYPE</td>
<td>2</td>
</tr>
<tr>
<td>BUILDINGID</td>
<td>GBRBOUR0001438</td>
</tr>
<tr>
<td>AREA2D_m2</td>
<td>212.832417013</td>
</tr>
<tr>
<td>AREA3D_m2</td>
<td>295.212341936</td>
</tr>
<tr>
<td>AZIMUTH</td>
<td>255.528377831</td>
</tr>
<tr>
<td>Area</td>
<td>212.832417013</td>
</tr>
<tr>
<td>BLOCKID</td>
<td>6</td>
</tr>
<tr>
<td>BUILDINGID</td>
<td>GBRBOUR0001438</td>
</tr>
<tr>
<td>CO2_REDUCTION</td>
<td>24.0594639109</td>
</tr>
<tr>
<td>ENERGY</td>
<td>44212.3266399</td>
</tr>
<tr>
<td>FLATROOF</td>
<td>0</td>
</tr>
<tr>
<td>INCLINATION</td>
<td>43.8672374622</td>
</tr>
<tr>
<td>IRR_GLOBAL</td>
<td>1101.20950542</td>
</tr>
</tbody>
</table>
### Basics statistics

**Input Vector Layer**
- irradiation

**Target field**
- IRR_GLOBAL

**Statistics output**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>867.304618585</td>
</tr>
<tr>
<td>StdDev</td>
<td>177.31912193</td>
</tr>
<tr>
<td>Sum</td>
<td>5952311.59735</td>
</tr>
<tr>
<td>Min</td>
<td>0.0</td>
</tr>
<tr>
<td>Max</td>
<td>1151.46218065</td>
</tr>
<tr>
<td>N</td>
<td>6863.0</td>
</tr>
<tr>
<td>CV</td>
<td>0.20444849264</td>
</tr>
<tr>
<td>Number of unique values</td>
<td>6853</td>
</tr>
<tr>
<td>Range</td>
<td>1151.46218065</td>
</tr>
<tr>
<td>Median</td>
<td>884.736791667</td>
</tr>
</tbody>
</table>

Press Ctrl+C to copy results to the clipboard

© Arup 2011

Statistics from GIS model
Web interface experiments - Flex
{ "type": "FeatureCollection", "features": [
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 1, "ROOFTYPE": 2, "ROOFSUB": 1, "REPAIR": 0, "Area": 7.103009 }, "geometry": { "type": "Polygon", "coordinates": [[ -208526.475416, 6575233.842579 ], [ -208526.965916, 6575231.910008 ], [ -208535.579035, 6575234.108296 ], [ -208535.088539, 6575236.040709 ], [ -208526.475416, 6575233.842579 ]] } }
,
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 2, "ROOFTYPE": 2, "ROOFSUB": 1, "REPAIR": 0, "Area": 2.748727 }, "geometry": { "type": "Polygon", "coordinates": [[ -208519.872014, 6575232.157297 ], [ -208517.506644, 6575233.361556 ], [ -208516.038868, 6575231.736187 ], [ -208519.482283, 6575230.000100 ], [ -208519.872014, 6575232.157297 ]] } }
,
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 2, "ROOFTYPE": 2, "ROOFSUB": 1, "REPAIR": 0, "Area": 5.808792 }, "geometry": { "type": "Polygon", "coordinates": [[ -208519.872014, 6575232.157297 ], [ -208517.506644, 6575233.361556 ], [ -208516.038868, 6575231.736187 ], [ -208519.482283, 6575230.000100 ], [ -208519.872014, 6575232.157297 ]] } }
,
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 3, "ROOFTYPE": 3, "ROOFSUB": 1, "REPAIR": 0, "Area": 3.901491 }, "geometry": { "type": "Polygon", "coordinates": [[ -208514.575594, 6575237.500646 ], [ -208516.038868, 6575231.736187 ], [ -208517.506644, 6575233.361556 ], [ -208516.341564, 6575237.951251 ], [ -208514.575594, 6575237.500646 ]] } }
,
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 3, "ROOFTYPE": 3, "ROOFSUB": 1, "REPAIR": 0, "Area": 24.153967 }, "geometry": { "type": "Polygon", "coordinates": [[ -208531.171831, 6575248.206274 ], [ -208525.177928, 6575242.341525 ], [ -208517.261731, 6575244.656177 ], [ -208514.799085, 6575244.027724 ], [ -208523.197967, 6575239.588700 ], [ -208533.519228, 6575242.222763 ], [ -208531.171831, 6575248.206274 ]] } }
,
    { "type": "Feature", "properties": { "BUILDINGID": "GBRBOUR0001395", "BLOCKID": 3, "ROOFTYPE": 3, "ROOFSUB": 1, "REPAIR": 0, "Area": 15.543203 }, "geometry": { "type": "Polygon", "coordinates": [[ -208523.197967, 6575239.588700 ], [ -208514.799085, 6575244.027724 ], [ -208516.341564, 6575237.951251 ], [ -208517.506644, 6575233.361556 ], [ -208523.197967, 6575239.588700 ]] } }
]}

geoJSON
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>OGR2Layers</title>
<style>
#map{width:400px;height:400px;}
</style>
<script src="http://www.openlayers.org/api/OpenLayers.js"></script>
<script type="text/javascript">
var map, selectsControls
function init(){
    var option = {
        projection: new OpenLayers.Projection("EPSG:900913"),
        displayProjection: new OpenLayers.Projection("EPSG:4326")
    };
    map = new OpenLayers.Map('map', option);
    olmapnik = new OpenLayers.Layer.OSM("OpenStreetMap Mapnik", "http://tile.openstreetmap.org/${z}/${x}/${y}.png");
    map.addLayer(olmapnik);
    map.setBaseLayer(olmapnik);
    var ls= new OpenLayers.Control.LayerSwitcher();
    map.addControl(ls);
    ls.maximizeControl();
    map.addControl(new OpenLayers.Control.MousePosition());
    map.addControl(new OpenLayers.Control.Scale());
    map.addControl(new OpenLayers.Control.Permalink());
    map.addControl(new OpenLayers.Control.Attribution());
    map.addControl(new OpenLayers.Control.OverviewMap());
    map.addControl(new OpenLayers.Control.PanZoomBar());
    var roofs_template = {
        strokeColor: "#000000",
        strokeOpacity: 1,
        strokeWidth: 0.26,
        fillColor: "#0122c1",
        fillOpacity: 1
    }
}
var roofs_style = new OpenLayers.Style(roofs_template)

    //START QUERY roofs
    function onPopupCloseroofs(evt) {
        selectControl.unselect(selectedFeature);
    }
    function onFeatureSelectroofs(feature) {
        selectedFeature = feature;
        tableroofs = "<html><meta http-equiv='Content-Type' content='text/html; charset=UTF-8'><body><table><tr><td><b>BUILDINGID:</b></td><td><i>"+feature.attributes.BUILDINGID+'</i></td></tr><tr><td><b>BLOCKID:</b></td><td><i>"+feature.attributes.BLOCKID+'</i></td></tr><tr><td><b>ROOFTYPE:</b></td><td><i>"+feature.attributes.ROOFTYPE+'</i></td></tr><tr><td><b>ROOFSUB:</b></td><td><i>"+feature.attributes.ROOFSUB+'</i></td></tr><tr><td><b>REPAIR:</b></td><td><i>"+feature.attributes.REPAIR+'</i></td></tr><tr><td><b>Area:</b></td><td><i>"+feature.attributes.Area+'</i></td></tr></table></body></html>";
        popup = new OpenLayers.Popup.FramedCloud("popup",
            feature.geometry.getBounds().getCenterLonLat(),
            new OpenLayers.Size(1000,500),
            tableroofs,
            null,
            true,
            onPopupCloseroofs
        );
        feature.popup = popup;
        map.addPopup(popup);
    }
    function onFeatureUnselectroofs(feature) {
        map.removePopup(feature.popup);
        feature.popup.destroy();
        feature.popup = null;
    }
map.addLayer(roofs);
selectControl = new OpenLayers.Control.SelectFeature(
    [roofs, ],
    {
        clickout: true, toggle: false,
        multiple: false, hover: false,
        toggleKey: "ctrlKey", // ctrl key removes from selection
        multipleKey: "shiftKey" // shift key adds to selection
    }
);
map.addControl(selectControl);
selectControl.activate();
roofs.events.on(
    {
        "featureselected": function(e) {
            onFeatureSelectroofs(e.feature);
        },
        "featureunselected": function(e) {
            onFeatureUnselectroofs(e.feature);
        }
    });
map.zoomToExtent(extent);
Arup Irradiation Mapping Test - Bournemouth
Arup Irradiation Mapping Test - Bournemouth
Thank you!

Francesco Anselmo
francesco.anselmo@arup.com