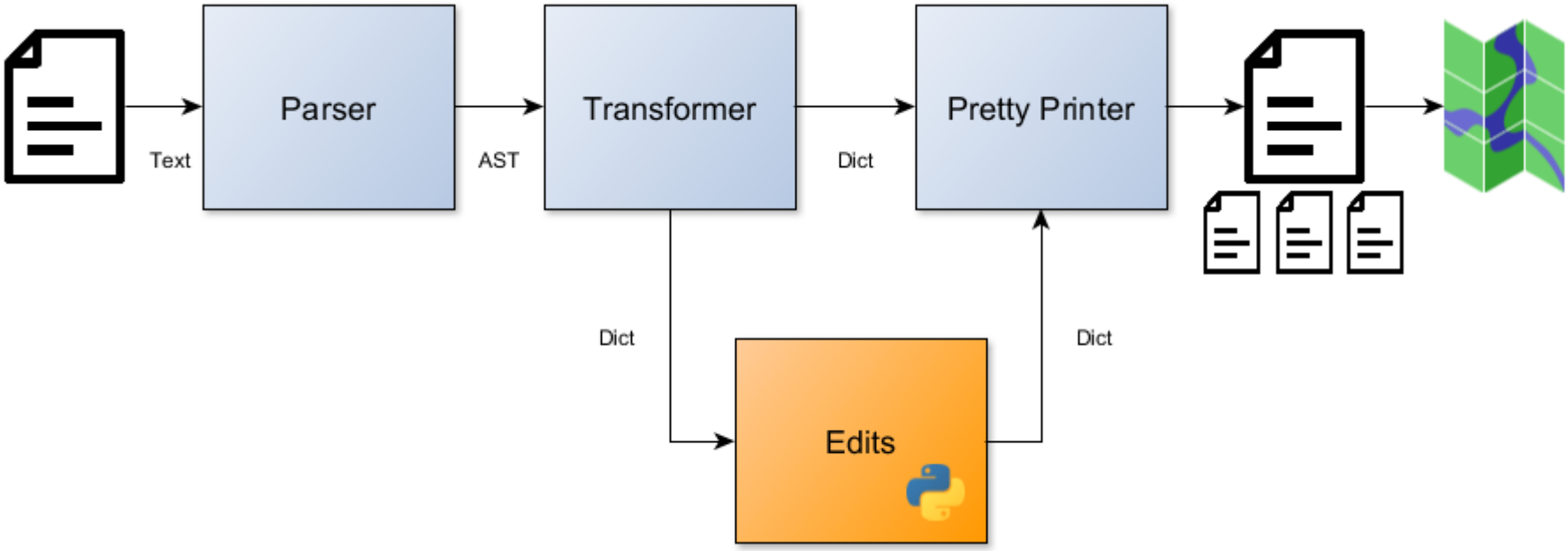
Introducing mappyfile

Seth Girvin

Mappyfile overview



- Parse, manipulate, and format Mapfiles
- Open-source license (MIT)
- Python 2 and 3 compatible

MapServer

open source web mapping

> MapServer is a rendering engine for beautiful maps

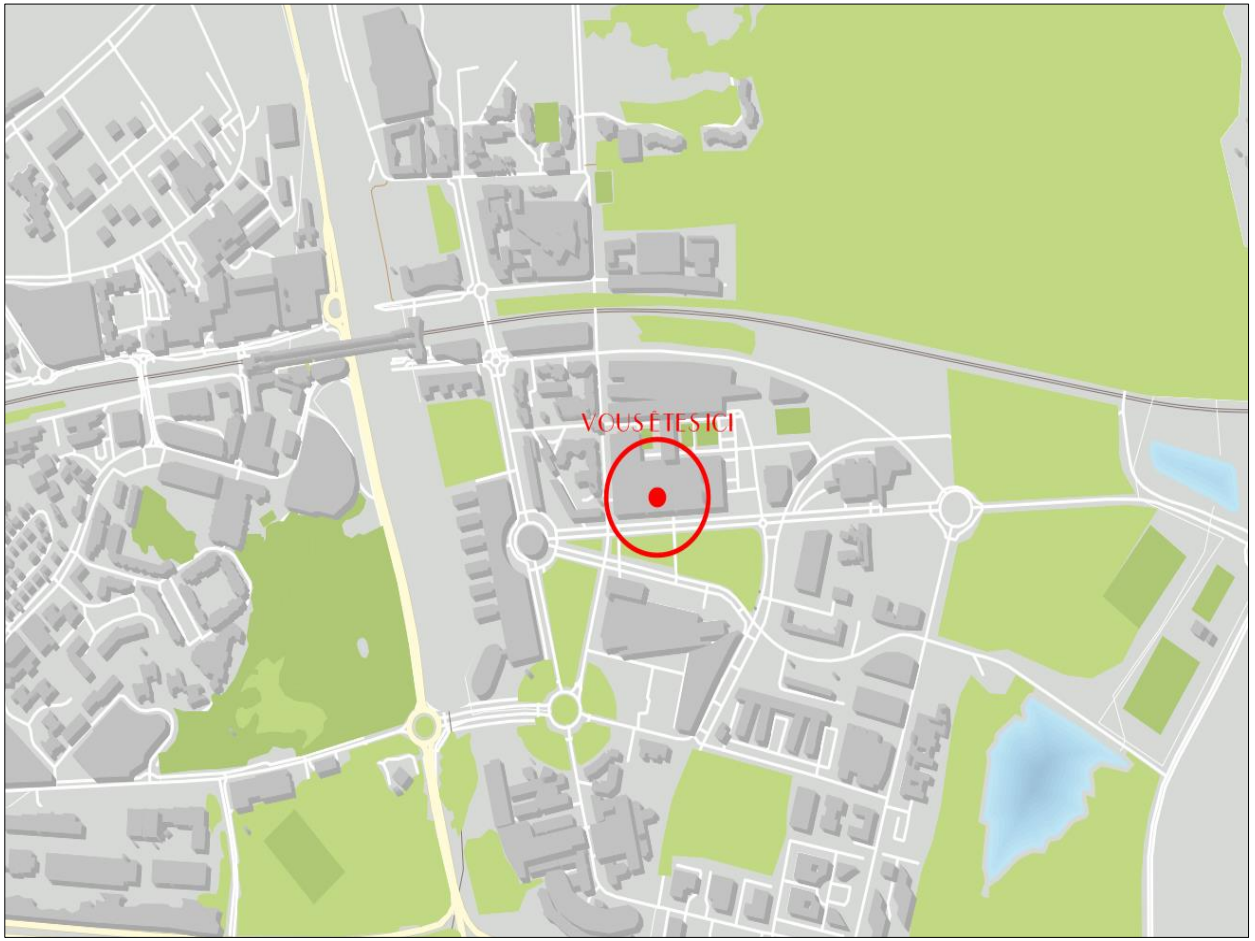
Paul Ramsey, PostGIS founder

Created in 1994

Built on top of libraries such as GDAL, AGG, GEOS

Fast and

Powerful



Two ways to access MapServer functionality

1. Via the CGI command line application, using either:
    - Mapfile
    - Runtime variables, to alter portions of a Mapfile via a query string

```
&map.layer[lakes].class[0].style[0]=SYMBOL+crosshatch+COLOR+151+51+151+SIZE+15&
```

2. Via MapScript a universal API generated by SWIG
    - Python, PHP, C#

> MapServer is first and foremost a CGI application, and Mapfile is the real specification of its usage - not MapScript.
>
> **Howard Butler (MapServer Project Steering Committee)**

- Feature lag between Mapfile and MapScript
- Window's binaries must be compiled in Microsoft's Visual C compiler 2008
- MapScript install issues
- Incomplete Python 3 support
- Much of application code moved to client-side libraries

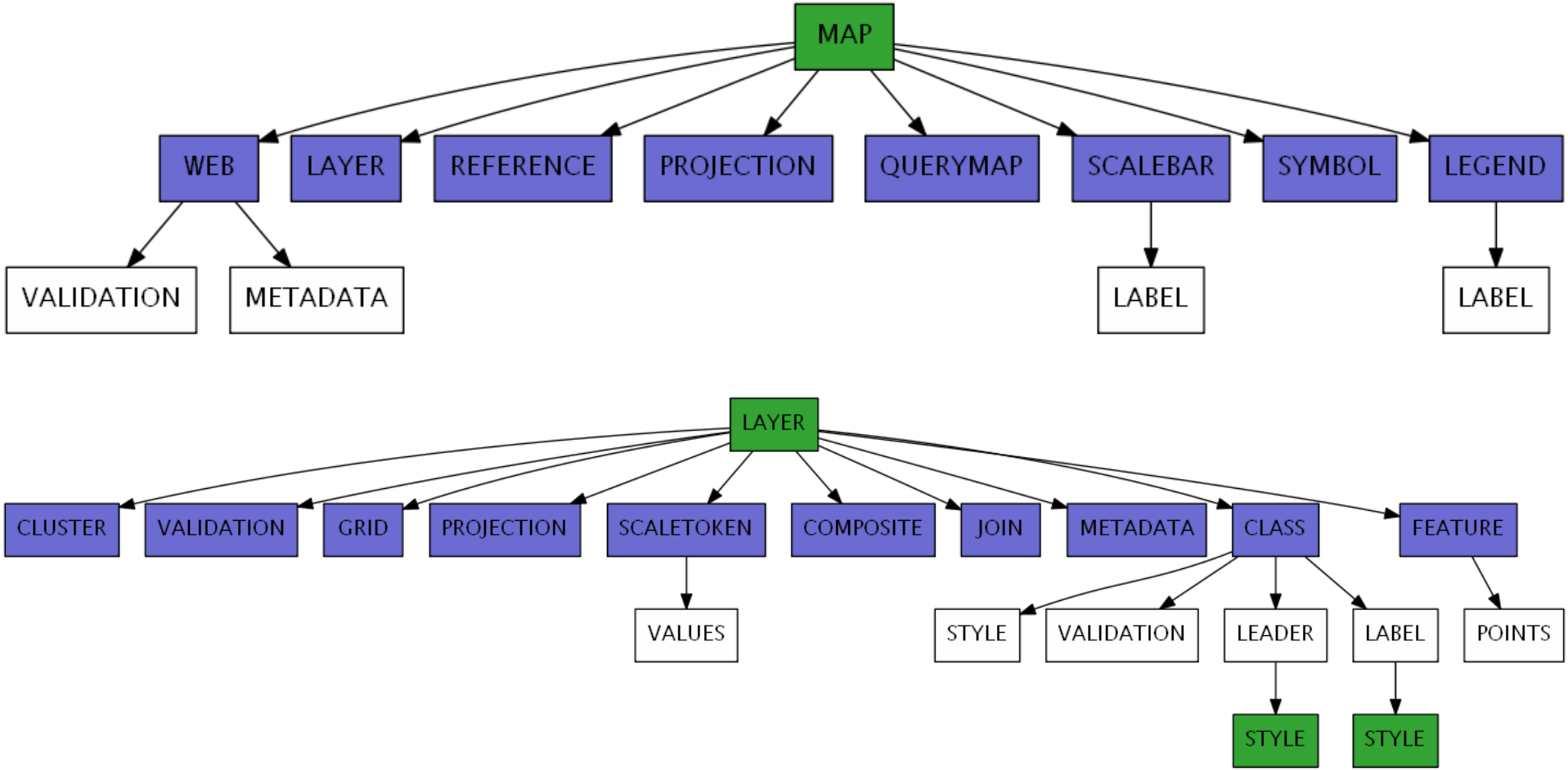
- *loadFromString - load Mapfile fragments in to various MapScript objects

Mapfile a Domain Specific Language (DSL)

> The instructions encoded in a MapServer Mapfile comprise a domain-specific language.. to embrace the map language is to benefit from simplicity, usability, and portability.

Sean Gillies, former MapScript maintainer

```
MAP
    WEB
        METADATA
            "wms_enable_request" "*"
        END
    END
    PROJECTION
        "init=epsg:4326"
    END
    LAYER
        NAME "land"
        TYPE POLYGON
        DATA "../data/vector/naturalearth/ne_110m_land"
        CLASS
            STYLE
                COLOR 107 208 107
                OUTLINECOLOR 2 2 2
                WIDTH 1
            END
        END
    END
END
```



- Declarative
- Mirrors the structure of a map
- Easier to learn with for non-programmers
- More flexible than a GUI

Syntax Examples

```
MAP
    LAYER
        NAME 'test' # key value
    END
END

MAP
    PROJECTION
        AUTO # single value
    END
END

CLASS
  EXPRESSION ( [EPPL_Q100_] = %eppl% ) # expressions
END

MAP
    CONFIG "PROJ_LIB" "projections" # three values
END

MAP CLASS NAME 'Test' STYLE OUTLINECOLOR 0 0 0 END END END

SYMBOL NAME 'triangle' TYPE VECTOR FILLED TRUE POINTS 0 4 2 0 4 4 0 4 END END
```

erezsh / lark

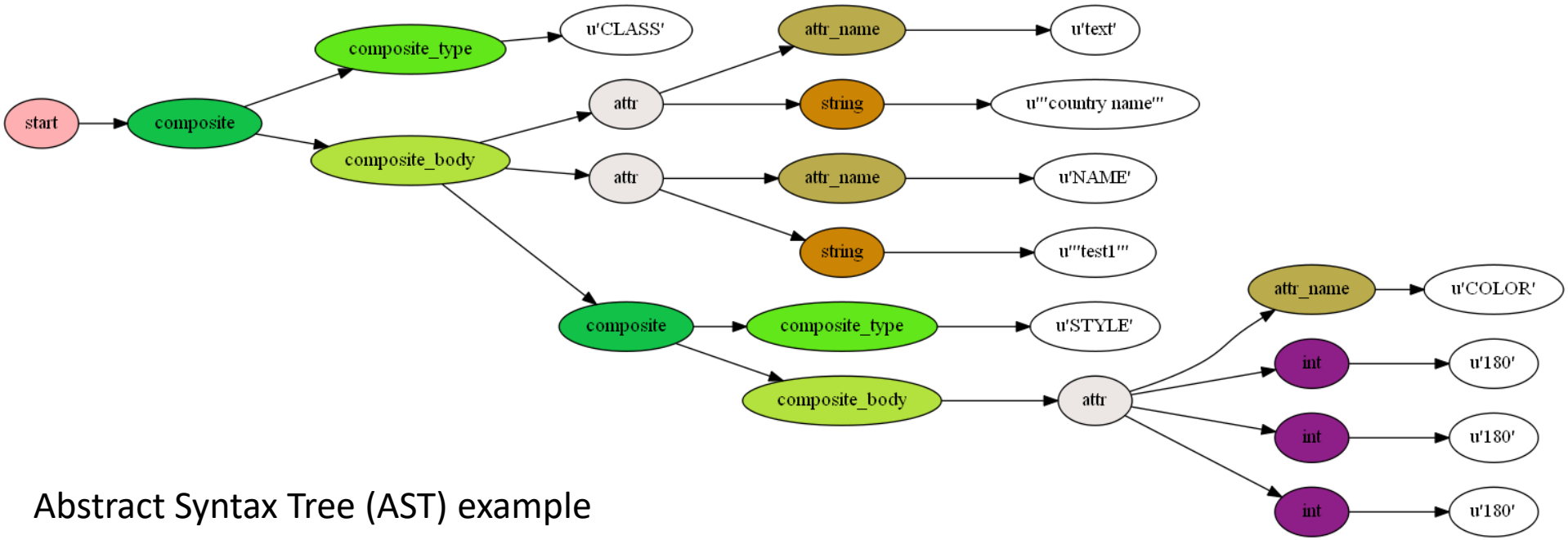
Unwatch 17 | Unstar 152 | Fork 9

Code | Issues 5 | Pull requests 0 | Projects 0 | Wiki | Insights

A modern parsing library for Python, implementing Earley & LALR(1) and an easy interface

parser | parsing-library | parser-library | parsing-engine | earley | lalr

Lark – created by Erez Shinan



Abstract Syntax Tree (AST) example

Fruit flies like a banana

ambiguities in grammar..

```
FEATURE
    POINTS
        1 1
        50 50
        1 50
        1 1
    END
END
```

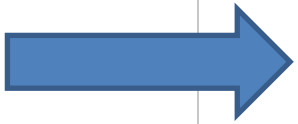
```
OUTPUTFORMAT
    NAME "shapezip"
    DRIVER "OGR/ESRI Shapefile"
    TRANSPARENT FALSE
    IMAGEMODE FEATURE
END
```

```
start: (_NL* composite _NL*)+

composite: composite_type attr? _NL+ composite_body _END
       | composite_type points _END
       | composite_type pattern _END
       | composite_type attr _END
           | metadata
           | validation

composite_body: _composite_item*
_composite_item: (composite|attr|points|projection|pattern|values) _NL+

points: "POINTS"i _NL* (_num_pair _NL*)* _END
pattern: "PATTERN"i _NL* (_num_pair _NL*)* _END

projection: "PROJECTION"i _NL* ((string _NL*)+|AUTO _NL+) _END
values: "VALUES"i _NL* ((string_pair) _NL+)+ _END

metadata: "METADATA"i _NL* ((string_pair|attr) _NL+)+ _END
validation: "VALIDATION"i _NL* ((string_pair|attr) _NL+)+ _END

attr: attr_name value+

attr_name: NAME | composite_type
?value: bare_string | string | int | float | expression | not_expression | attr_bind | path | regexp | runtime_var | list

int: SIGNED_INT
int_pair: int int
!bare_string: NAME | "SYMBOL"i | "AUTO"i | "GRID"i | "CLASS"i | "FEATURE"i
string: STRING1 | STRING2 | STRING3
string_pair: string string
float: SIGNED_FLOAT
float_pair: float float
path: PATH
regexp: REGEXP1 | REGEXP2
runtime_var: RUNTIME_VAR
list: "{" value ("," value)* "}"

_num_pair: (int|float) _NL* (int|float)

attr_bind: "[" bare_string "]"

not_expression: ("!"|"NOT"i) expression
expression: "(" or_test ")"
?or_test : (or_test ("OR"i|"||"))? and_test
?and_test : (and_test ("AND"i|"&&"))? comparison
?comparison: (comparison compare_op)? add
!compare_op: ">=" | "<" | "=*" | "==" | "=" | "~" | "~*" | ">" | "<=" | "IN" | "NE" | "EQ"

?add: (add "+")? (func_call | value)
func_call: attr_name "(" func_params ")"
func_params: value ("," value)*

!composite_type: "CLASS"i
            | "CLUSTER"i
            | "COMPOSITE"i
            | "CONFIG"i
            | "FEATURE"i
            | "FONTSET"i
            | "GRID"i
            | "INCLUDE"i
            | "JOIN"i
            | "LABEL"i
            | "LAYER"i
            | "LEADER"i
            | "LEGEND"i
            | "MAP"i
            | "OUTPUTFORMAT"i
            | "QUERYMAP"i
            | "REFERENCE"i
            | "SCALEBAR"i
            | "SCALETOKEN"i
            | "STYLE"i
            | "SYMBOL"i
            | "WEB"i

AUTO: "AUTO"i
PATH: /[a-z_]*[.\/][a-z0-9_\/.]+/i
NAME: /[a-z_][a-z0-9_]*/i

SIGNED_FLOAT: ["-"|"+"] FLOAT
SIGNED_INT: ["-"|"+"] INT

%import common.FLOAT
%import common.INT

STRING1: /".*?(?<!\\)(\\\\)*?"i?/
STRING2: /'.*?(?<!\\)(\\\\)*?'i?/
STRING3: /`.*?`i?/   // XXX TODO
REGEXP1: /\/.*?\/i?/
REGEXP2: /\\\.*?\\\i?/
RUNTIME_VAR: /%.*?%/

COMMENT: /\#[^\n]*/
CCOMMENT: /\/(?s)[*].*?[*]\//

_END: "END"i

WS: /[ \t\f]+/
_NL: /[\r\n]+/

%ignore COMMENT
%ignore CCOMMENT
%ignore WS
```

The complete Grammar file

Transform to dictionary

```
MAP
    NAME "sample"
    STATUS ON
    SIZE 600 400
    SYMBOLSET "../etc/symbols.txt"
    EXTENT -180 -90 180 90
    UNITS DD
    SHAPEPATH "../data"
    IMAGECOLOR 255 255 255
    FONTSET "../etc/fonts.txt"

    #
    # Start of web interface definition
    #
    WEB
        IMAGEPATH "/ms4w/tmp/ms_tmp/"
        IMAGEURL "/ms_tmp/"
    END # WEB

    #
    # Start of layer definitions
    #
    LAYER
        NAME 'global-raster'
        TYPE RASTER
        STATUS DEFAULT
        DATA bluemarble.gif
    END # LAYER
END # MAP
```



```
{
    "name": "sample",
    "status": "ON",
    "size": [
        600,
        400
    ],
    "symbolset": "../etc/symbols.txt",
    "extent": [
        -180,
        -90,
        180,
        90
    ],
    "units": "DD",
    "shapepath": "../data",
    "imagecolor": [
        255,
        255,
        255
    ],
    "fontset": "../etc/fonts.txt",
    "web": {
        "imagepath": "/ms4w/tmp/ms_tmp/",
        "imageurl": "/ms_tmp/",
        "__type__": "web"
    },
    "layers": [
        {
            "name": "'global-raster'",
            "type": "RASTER",
            "status": "DEFAULT",
            "data": "bluemarble.gif",
            "__type__": "layer"
        }
    ],
    "__type__": "map"
}
```

msautotest

770 Mapfiles: Parse, output, reparse

Pretty Printing

mappyfile.geographika.net

mappyfile

A Python Mapfile parser for MapServer

pypi v0.3.1 | build passing | docs latest

Indent 4

Quotes Double

Please avoid pasting any sensitive data such as passwords in case they end up in web server logs

Format

```
MAP
    NAME "sample"
    STATUS ON
    SIZE 600 400
    SYMBOLSET "../etc/symbols.txt"
    EXTENT -180 -90 180 90
    UNITS DD
    SHAPEPATH "../data"
    IMAGECOLOR 255 255 255
    FONTSET "../etc/fonts.txt"
    WEB
        IMAGEPATH "/ms4w/tmp/ms_tmp/"
        IMAGEURL "/ms_tmp/"
    END
    LAYER
        NAME "global-raster"
        TYPE RASTER
        STATUS DEFAULT
        DATA bluemarble.gif
    END
END
```

More Pythonic

No need to create **lots of objects** and worry about their **object lifetimes** and **relationships**

MapScript

```
# define class strings
c1 = """
CLASS
    NAME 'The World'
    STYLE
        OUTLINECOLOR 0 255 0
    END
END"""

c2 = """
CLASS
    NAME 'Roads'
    STYLE
        OUTLINECOLOR 0 0 0
    END
END"""

# remove existing classes
for idx in reversed(range(0, layer.numclasses)):
    layer.removeClass(idx)

# create a new class object from the strings and add to the layer
for c in classes:
    clsObj = mapscript.fromstring(c)
    layer.classes.append(clsObj)
```

mappyfile

```
# define all classes in a single string
classes = """
CLASS
    NAME 'The World'
    STYLE
        OUTLINECOLOR 0 255 0
    END
END
CLASS
    NAME 'Roads'
    STYLE
        OUTLINECOLOR 0 0 0
    END
END
"""

# parse the string and replace the existing classes for the layer
layer["classes"] = mappyfile.loads(classes)
```

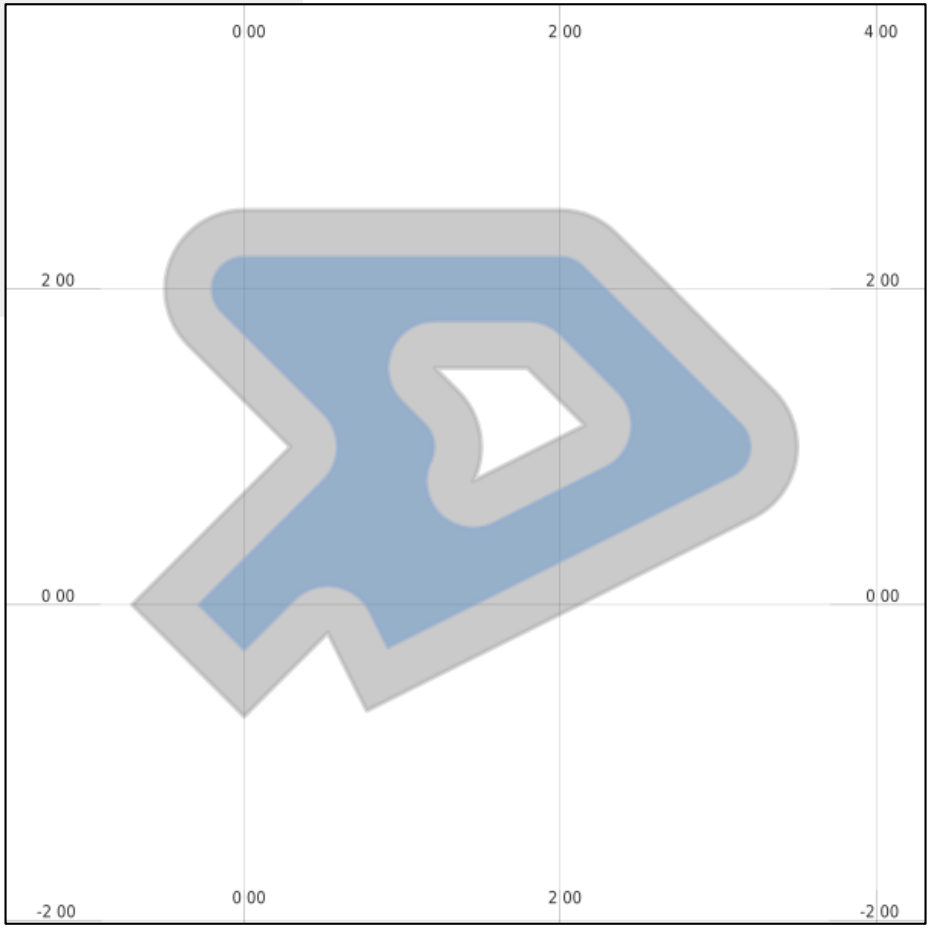
Uses

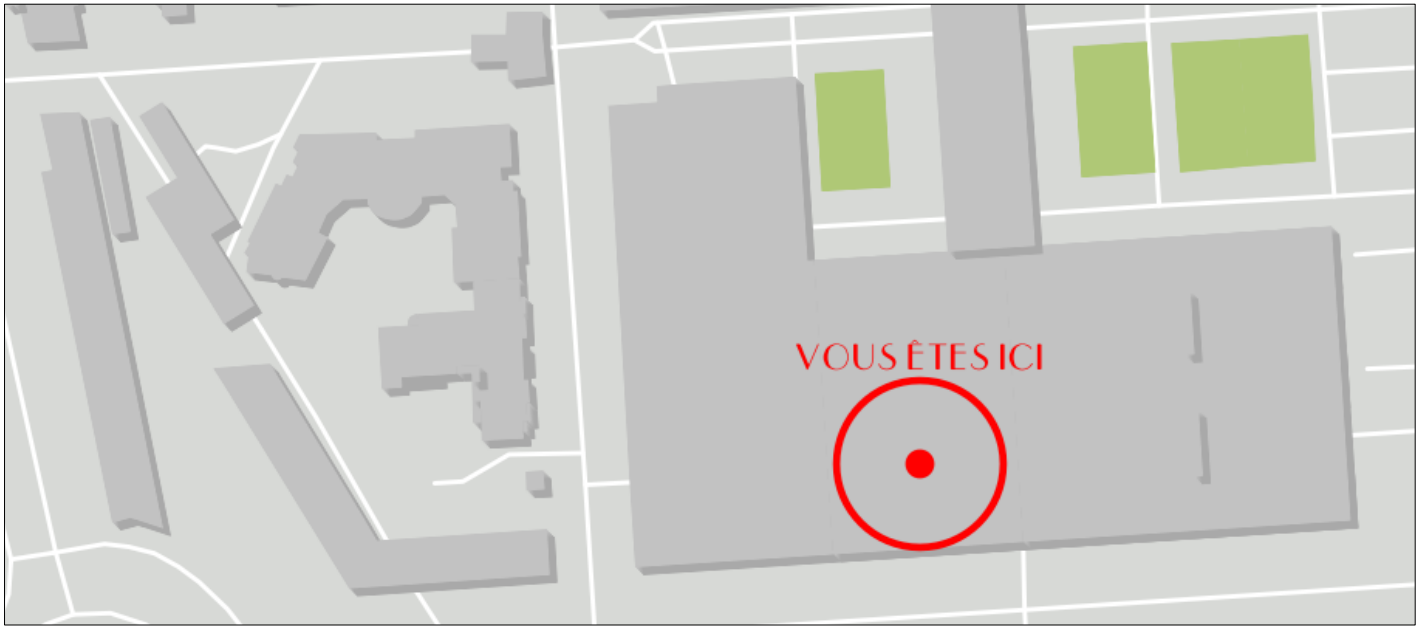
- Create maps as part of a Python workflow
- Create client specific maps from a master map
- Easily add “boiler-plate code” – save copy and pasting
- Dynamic styling through a web service

```python
def erosion(mapfile, dilated):
    ll = mappyfile.find(mapfile["layers"], "name", "line")
    ll["status"] = "OFF"

    pl = mappyfile.find(mapfile["layers"], "name", "polygon")

    # make a deep copy of the polygon layer in the Map
    # so any modification are made to this layer only
    pl2 = deepcopy(pl)

    pl2["name"] = "newpolygon"
    mapfile["layers"].append(pl2)

    dilated = dilated.buffer(-0.3)
    pl2["features"][0]["wkt"] = "'%s'" % dilated.wkt

    style = pl["classes"][0]["styles"][0]
    style["color"] = "'#999999'"
    style["outlinecolor"] = "'#b2b2b2'"
```

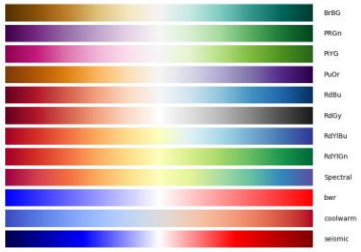
python powered

Geometry rendering





colormap



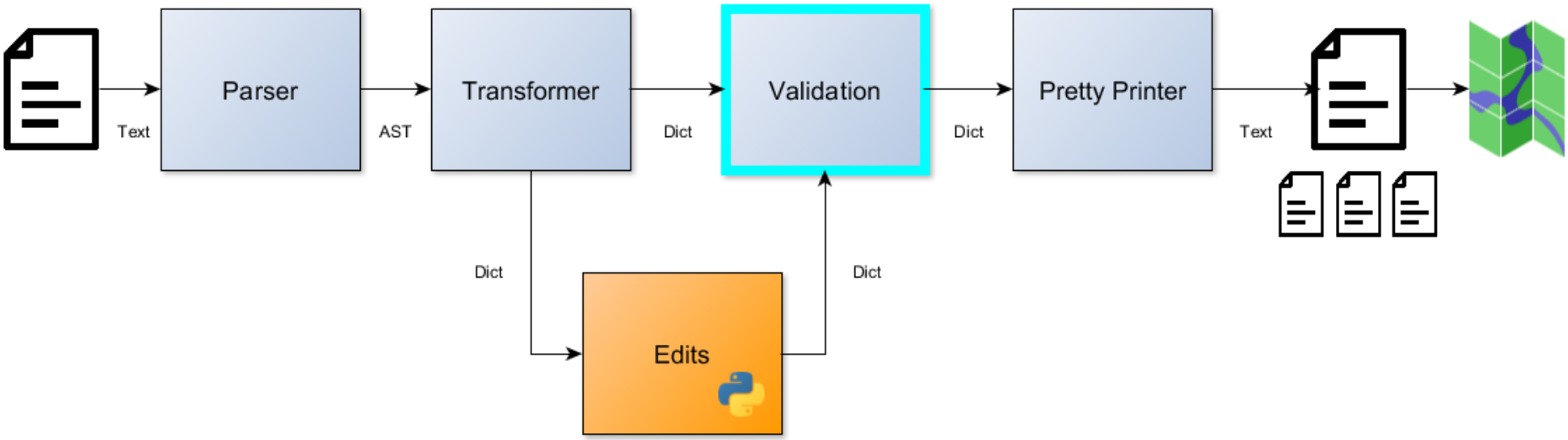

```
CLASS
    EXPRESSION ((((( "[natural]" = 'water' ))))
    NAME Water
    STYLE
        COLOR 192 230 246
        GEOMTRANSFORM ((((([shape],-5))))))
    END
    STYLE
        COLOR 187 226 243
        GEOMTRANSFORM ((((([shape],-10))))))
    END
    STYLE
        COLOR 182 222 241
        GEOMTRANSFORM ((((([shape],-15))))))
    END
    STYLE
        COLOR 178 218 238
        GEOMTRANSFORM ((((([shape],-20))))))
    END
    STYLE
        COLOR 174 214 236
        GEOMTRANSFORM ((((([shape],-25))))))
    END
    STYLE
        COLOR 170 210 233
        GEOMTRANSFORM ((((([shape],-30))))))
    END
    STYLE
        COLOR 166 206 229
        GEOMTRANSFORM ((((([shape],-35))))))
    END
```


```
LAYER
    NAME "Settlement_Buffers_1000"
    TYPE POINT
    CONNECTIONTYPE ogr
    CONNECTION "settlements.gdb"
    DATA "Settlement_Centroids"
    SIZEUNITS meters
    PROJECTION
        'init=epsg:2157'
    END
    CLASS
        STYLE
            SIZE 1000
            COLOR "#33ccff"
            SYMBOL "CIRCLE"
            OUTLINECOLOR "#00394d"
            OPACITY 30
        END
    END
END
LAYER
    NAME "Settlement_Buffers_500"
    TYPE POINT
    CONNECTIONTYPE ogr
    CONNECTION "settlements.gdb"
    DATA "Settlement_Centroids"
    SIZEUNITS meters
    PROJECTION
        'init=epsg:2157'
    END
    CLASS
        STYLE
            SIZE 500
            COLOR "#33ccff"
            SYMBOL "CIRCLE"
            OUTLINECOLOR "#00394d"
            OPACITY 60
        END
    END
END
LAYER
    NAME "Settlement_Buffers_250"
    TYPE POINT
```

Buffering

Add **Validation**

- A *linter* for Mapfiles
- Highlight deprecated keywords
- Report errors



- Improve performance

Thanks for listening!

https://github.com/geographika/mappyfile

http://mappyfile.readthedocs.io

https://pypi.python.org/pypi/mappyfile

```
pip install mappyfile
```

Thanks to:

- Erez Shinan for the mappyfile grammar
- All the MapServer developers for 23 years of development

sgirvin@compass.ie
@geographika