

Table of Contents

Layer	1
Basics	1
Geoprocessing	22
Graticule	83

Layer

Basics

Open

Open a Layer.

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
layer	The Layer name	true		
name	The name	false		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> workspace close --name naturalearth  
Workspace naturalearth closed!
```

Close

Close a Layer.

```
geo-shell> layer close --name countries
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer close --name countries  
Layer countries closed!
```

```
geo-shell> workspace close --name naturalearth  
Workspace naturalearth closed!
```

List

List open Layers.

```
geo-shell> layer list
```



No parameters

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states  
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer list  
countries = GeoPackage  
ocean = GeoPackage  
states = GeoPackage
```

```
geo-shell> workspace close --name naturalearth  
Workspace naturalearth closed!
```

Schema

Inspect a Layer's Schema.

```
geo-shell> layer schema --name countries
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer schema --name countries  
Name Type
```

```
-----  
the_geom MultiPolygon  
ScaleRank Integer  
FeatureCla String
```

SOVEREIGNT String
SOVISO String
SOV_A3 String
LEVEL Double
TYPE String
NAME String
SORTNAME String
ADM0_A3 String
NAME_SM String
NAME_LNG String
TERR_ String
PARENTHETI String
NAME_ALT String
LOCAL_LNG String
LOCAL_SM String
FORMER String
ABBREV_ String
MAP_COLOR Double
PEOPLE Double
GDP_USDM Double
FIPS_10 String
ISO_A2 String
ISO_A3 String
ISO_N3 Double
ITU String
IOC String
FIFA String
DS String
WMO String
GAUL Double
MARC String
STANAG1059 String
GW_ID Double
DIAL Double
INTERNET_ String
COG String
ACTUAL String
CAPAY String
CRPAY String
ANI String
LIBENR String
ANCNOM String
PAYS_R_GIO String
COMMENT String

geo-shell> **workspace close** --name naturalearth
Workspace naturalearth closed!

Count

Count the Feature in a Layer.

```
geo-shell> layer count --name countries
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer count --name countries  
177
```

```
geo-shell> workspace close --name naturalearth  
Workspace naturalearth closed!
```

Projection

Get the Projection of a Layer.

```
geo-shell> layer projection --name countries
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer projection --name countries  
EPSG:4326
```

```
geo-shell> workspace close --name naturalearth  
Workspace naturalearth closed!
```

Features

Display the Features of a Layer.

```
geo-shell> layer features --name states --filter "NAME_1='North Dakota'"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
filter	The CQL Filter	false		
sort	A Sort parameter (fld dir)	false		
start	The start index	false		-1
max	The maximum number of records	false		-1
field	A subfield to include	false		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer features --name states --filter "NAME_1='North Dakota'"
```

Feature (states.3)

```
-----
the_geom = MULTIPOLYGON
FID_1 = 31
ScaleRank = 2
FeatureCla = 1st Order Admin Polys
OBJECTID = 22
VertexCou = 223.0
ISO = USA
NAME_0 = United States
NAME_1 = North Dakota
VARNAME_1 = ND | N.D.
NL_NAME_1 =
HASC_1 = US.ND
TYPE_1 = State
ENGTYPE_1 = State
VALIDFR_1 = 18891102
VALIDTO_1 = Present
REMARKS_1 =
Region =
RegionVar =
ProvNumber = 23
NEV_Countr = United States
FIRST_FIPS =
FIRST_HASC =
FIPS_1 = US38
gadm_level = 1.0
```

```
CheckMe = 0
Region_Cod =
Region_C_1 =
ScaleRan_1 = 1
Region_C_2 =
Region_C_3 =
Country_Pr =
```

```
geo-shell> workspace close --name naturalearth
Workspace naturalearth closed!
```

Get Style

Get the Layer's style.

```
geo-shell> layer style get --name states --style target/states.sld
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
style	The SLD File	false		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> style vector default --layer states --color #1E90FF --file examples/states_simple.sld
Default Vector Style for states written to /home/travis/build/jericks/geo-shell/examples/states_simple.sld!
```

```
geo-shell> layer style get --name states --style target/states.sld
states style written to /home/travis/build/jericks/geo-shell/target/states.sld
```

```
geo-shell> workspace close --name naturalearth
Workspace naturalearth closed!
```

```
<?xml version="1.0" encoding="UTF-8"?><sld:StyledLayerDescriptor
xmlns="http://www.opengis.net/sld" xmlns:sld="http://www.opengis.net/sld"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
version="1.0.0">
  <sld:UserLayer>
    <sld:LayerFeatureConstraints>
      <sld:FeatureTypeConstraint/>
    </sld:LayerFeatureConstraints>
    <sld:UserStyle>
      <sld:Name>Default Styler</sld:Name>
      <sld:FeatureTypeStyle>
        <sld:Name>name</sld:Name>
        <sld:Rule>
          <sld:PolygonSymbolizer>
            <sld:Fill>
              <sld:CssParameter name="fill">#f2f2f2</sld:CssParameter>
            </sld:Fill>
          </sld:PolygonSymbolizer>
          <sld:LineSymbolizer>
            <sld:Stroke>
              <sld:CssParameter name="stroke">#a9a9a9</sld:CssParameter>
              <sld:CssParameter name="stroke-width">0.5</sld:CssParameter>
            </sld:Stroke>
          </sld:LineSymbolizer>
        </sld:Rule>
      </sld:FeatureTypeStyle>
    </sld:UserStyle>
  </sld:UserLayer>
</sld:StyledLayerDescriptor>
```

Set Style

Set a Layer's style

geo-shell> **layer style get** --name states --style target/states_simple.sld

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
style	The SLD or CSS File	true		

geo-shell> **workspace open** --name naturalearth --params src/test/resources/naturalearth.gpkg
Workspace naturalearth opened!

geo-shell> **layer open** --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states

geo-shell> **style vector default** --layer states --color #1E90FF --file examples/states_simple.sld

Default Vector Style for states written to /home/travis/build/jericks/geo-shell/examples/states_simple.sld!

geo-shell> **layer style get** --name states --style target/states_simple.sld
states style written to /home/travis/build/jericks/geo-shell/target/states_simple.sld

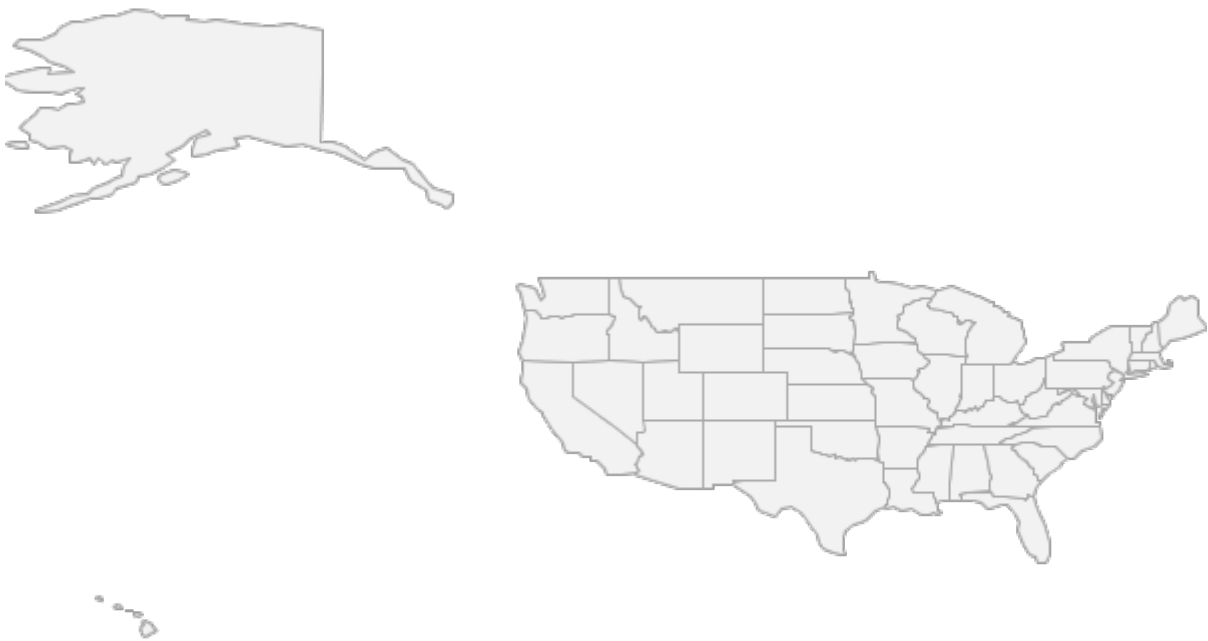
geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer states
Added states layer to map map

geo-shell> **map draw** --name map --file examples/layer_set_style.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_set_style.png!

geo-shell> **map close** --name map
Map map closed!

geo-shell> **workspace close** --name naturalearth
Workspace naturalearth closed!



Copy

Copy one Layer to another Workspace.

geo-shell> **layer copy** --input-name states_gpkg --output-workspace shapefiles --output-name states

Name	Description	Mandatory	Specified Default	Unspecified Default
------	-------------	-----------	-------------------	---------------------

input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
filter	The CQL Filter	false		
sort	A Sort parameter (fld dir)	false		
start	The start index	false		-1
max	The maximum number of records	false		-1
field	A subfield to include	false		

```
geo-shell> workspace open --name naturalearth --params src/test/resources/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states_gpkg
Opened Workspace naturalearth Layer states as states_gpkg
```

```
geo-shell> workspace open --name shapefiles --params target/
Workspace shapefiles opened!
```

```
geo-shell> layer copy --input-name states_gpkg --output-workspace shapefiles --output-name states
Done!
```

```
geo-shell> layer count --name states
52
```

```
geo-shell> workspace close --name shapefiles
Workspace shapefiles closed!
```

```
geo-shell> workspace close --name naturalearth
Workspace naturalearth closed!
```

Create

Create a new Layer.

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point
EPSG:4326|fid=Int|name=String"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		

name	The new Layer name	true		
fields	The pipe delimited list of fields (name=type)	true		

```
geo-shell> workspace open --name mem --params memory
Workspace mem opened!
```

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point
EPSG:4326 | fid=Int | name=String"
Created Layer points!
```

```
geo-shell> layer schema --name points
Name Type
```

```
-----
the_geom Point
fid Integer
name String
```

Add

Add a new Feature to a Layer.

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.333056
47.609722) | fid=1 | name=Seattle"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
values	The pipe delimited list of values (field=value)	true		

```
geo-shell> workspace open --name mem --params memory
Workspace mem opened!
```

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point
EPSG:4326 | fid=Int | name=String"
Created Layer points!
```

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.333056
47.609722) | fid=1 | name=Seattle"
Added Feature to points
```

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.459444
47.241389) | fid=2 | name=Tacoma"
Added Feature to points
```

```
geo-shell> layer count --name points
2
```

Delete

Delete features from the Layer

```
geo-shell> layer delete --name points --filter "fid=2"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
filter	The CQL Filter	true		

```
geo-shell> workspace open --name mem --params memory
Workspace mem opened!
```

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point
EPSG:4326 | fid=Int | name=String"
Created Layer points!
```

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.333056
47.609722) | fid=1 | name=Seattle"
Added Feature to points
```

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.459444
47.241389) | fid=2 | name=Tacoma"
Added Feature to points
```

```
geo-shell> layer count --name points
2
```

```
geo-shell> layer delete --name points --filter "fid=2"
Deleted fid=2 Features from points
```

```
geo-shell> layer count --name points
1
```

Remove

Remove a Layer from a Workspace.

```
geo-shell> layer remove --layer polygons --workspace mem
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
layer	The Layer name	true		

```
geo-shell> workspace open --name mem --params memory
```

Workspace mem opened!

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point  
EPSG:4326|fid=Int|name=String"
```

Created Layer points!

```
geo-shell> layer create --workspace mem --name lines --fields "the_geom=LineString  
EPSG:4326|fid=Int|name=String"
```

Created Layer lines!

```
geo-shell> layer create --workspace mem --name polygons --fields "the_geom=Polygon  
EPSG:4326|fid=Int|name=String"
```

Created Layer polygons!

```
geo-shell> workspace layers --name mem
```

lines

points

polygons

```
geo-shell> layer remove --layer polygons --workspace mem
```

Layer polygons removed from Workspace mem

```
geo-shell> workspace layers --name mem
```

lines

points

Update Field

Update the values of a field

```
geo-shell> layer updatefield --name points --field state --value WA
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
field	The field name	true		
value	The value	true		
filter	The CQL Filter	false	INCLUDE	INCLUDE
script	Whether the value is a script or not	false	false	false

```
geo-shell> workspace open --name mem --params memory
```

Workspace mem opened!

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point  
EPSG:4326|fid=Int|name=String|state=String"
```

Created Layer points!

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.333056 47.609722)|fid=1|name=Seattle"
```

Added Feature to points

```
geo-shell> layer add --name points --values "the_geom=POINT (-122.459444 47.241389)|fid=2|name=Tacoma"
```

Added Feature to points

```
geo-shell> layer updatefield --name points --field state --value WA  
Done updating state with WA!
```

```
geo-shell> layer features --name points
```

Feature (fid—34f3c37f_17060e5e693_-7890)

the_geom = POINT (-122.333056 47.609722)

fid = 1

name = Seattle

state = WA

Feature (fid—34f3c37f_17060e5e693_-788e)

the_geom = POINT (-122.459444 47.241389)

fid = 2

name = Tacoma

state = WA

Add Fields

Add Fields to the input Layer and save the result to the output Layer

```
geo-shell> layer addfields --input-name points --output-workspace mem --output-name points2  
--fields "name=String,state=String"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
fields	The Fields (name=type proj)	true		

```
geo-shell> workspace open --name mem --params memory  
Workspace mem opened!
```

```
geo-shell> layer create --workspace mem --name points --fields "the_geom=Point EPSG:4326"  
Created Layer points!
```

```
geo-shell> layer addfields --input-name points --output-workspace mem --output-name points2
--fields "name=String,state=String"
Done!
```

```
geo-shell> layer schema --name points2
Name Type
```

```
-----
the_geom Point
name String
state String
```

Add Area Field

Add area Field to the input Layer and save the result to the output Layer

```
geo-shell> layer addareafield --input-name states --output-workspace mem --output-name
states_area --area-fieldname AREA
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
area-fieldname	The area field name	true	area	area

```
geo-shell> workspace open --name mem --params memory
Workspace mem opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer addareafield --input-name states --output-workspace mem --output-name
states_area --area-fieldname AREA
Done!
```

```
geo-shell> layer schema --name states_area
Name Type
```

```
-----
the_geom MultiPolygon
FID_1 Integer
ScaleRank Integer
FeatureCla String
OBJECTID Integer
```

VertexCou Double
 ISO String
 NAME_0 String
 NAME_1 String
 VARNAME_1 String
 NL_NAME_1 String
 HASC_1 String
 TYPE_1 String
 ENGTYPE_1 String
 VALIDFR_1 String
 VALIDTO_1 String
 REMARKS_1 String
 Region String
 RegionVar String
 ProvNumber Integer
 NEV_Countr String
 FIRST_FIPS String
 FIRST_HASC String
 FIPS_1 String
 gadm_level Double
 CheckMe Integer
 Region_Cod String
 Region_C_1 String
 ScaleRan_1 Integer
 Region_C_2 String
 Region_C_3 String
 Country_Pr String
 AREA Double

```
geo-shell> layer features --name states_area --filter "NAME_1='North Dakota'" --field "NAME_0,AREA"
```

Feature (fid—34f3c37f_17060e5e693_-7889)

 NAME_0 = United States
 AREA = 21.804544852979944

Add ID Field

Add area ID to the input Layer and save the result to the output Layer

```
geo-shell> layer addidfield --input-name places --output-workspace mem --output-name places_id --id-fieldname ID --start-value 1
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		

output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
id-fieldname	The id field name	true	id	id
start-value	The value to start at	true	1	1

```
geo-shell> workspace open --name mem --params memory
Workspace mem opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer places --name places
Opened Workspace naturalearth Layer places as places
```

```
geo-shell> layer addidfield --input-name places --output-workspace mem --output-name places_id
--id-fieldname ID --start-value 1
Done!
```

```
geo-shell> layer schema --name places_id
Name Type
```

```
-----
the_geom Point
SCALERANK Integer
NATSCALE Integer
LABELRANK Integer
FEATURECLA String
NAME String
NAMEPAR String
NAMEALT String
DIFFASCII Integer
NAMEASCII String
ADM0CAP Double
CAPALT Double
CAPIN String
WORLDCITY Double
MEGACITY Integer
SOV0NAME String
SOV_A3 String
ADM0NAME String
ADM0_A3 String
ADM1NAME String
ISO_A2 String
NOTE String
LATITUDE Double
LONGITUDE Double
```

CHANGED Double
NAMEDIFF Integer
DIFFNOTE String
POP_MAX Integer
POP_MIN Integer
POP_OTHER Integer
GEONAMEID Double
MEGANAME String
LS_NAME String
LS_MATCH Integer
CHECKME Integer
MAX_POP10 Integer
MAX_POP20 Integer
MAX_POP50 Integer
MAX_POP300 Integer
MAX_POP310 Integer
MAX_NATSCA Integer
MIN_AREAKM Integer
MAX_AREAKM Double
MIN_AREAMI Double
MAX_AREAMI Double
MIN_PERKM Double
MAX_PERKM Double
MIN_PERMI Double
MAX_PERMI Double
MIN_BBXMING Double
MAX_BBXMING Double
MIN_BBXMING Double
MAX_BBXMING Double
MIN_BBYMING Double
MAX_BBYMING Double
MIN_BBYMING Double
MAX_BBYMING Double
MEAN_BBXC Double
MEAN_BBYC Double
COMPARE Integer
GN_ASCII String
FEATURE_CL String
FEATURE_CO String
ADMIN1_COD Double
GN_POP Integer
ELEVATION Double
GTOPO30 Double
TIMEZONE String
GEONAMESNO String
UN_FID Integer
UN_ADM0 String
UN_LAT Double

UN_LONG Double
 POP1950 Double
 POP1955 Double
 POP1960 Double
 POP1965 Double
 POP1970 Double
 POP1975 Double
 POP1980 Double
 POP1985 Double
 POP1990 Double
 POP1995 Double
 POP2000 Double
 POP2005 Double
 POP2010 Double
 POP2015 Double
 POP2020 Double
 POP2025 Double
 POP2050 Double
 CITYALT String
 popDiff Integer
 popPerc Double
 ls_gross Integer
 ID Integer

```
geo-shell> layer features --name places_id --filter "NAME='Seattle'" --field "NAME,ID"
```

Feature (fid—34f3c37f_17060e5e693_-79ce)

 NAME = Seattle
 ID = 10

Add XY Fields

Add x and y coordinate Fields to the input Layer and save the result to the output Layer

```
geo-shell> layer addxyfields --input-name places --output-workspace mem --output-name  
places_xy --x-fieldname X --y-fieldname Y
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
x-fieldname	The x field name	true	x	x
y-fieldname	The y field name	true	y	y

```
geo-shell> workspace open --name mem --params memory
```

Workspace mem opened!

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
```

Workspace naturalearth opened!

```
geo-shell> layer open --workspace naturalearth --layer places --name places
```

Opened Workspace naturalearth Layer places as places

```
geo-shell> layer addxyfields --input-name places --output-workspace mem --output-name  
places_xy --x-fieldname X --y-fieldname Y
```

Done!

```
geo-shell> layer schema --name places_xy
```

Name Type

```
-----  
the_geom Point  
SCALERANK Integer  
NATSCALE Integer  
LABELRANK Integer  
FEATURECLA String  
NAME String  
NAMEPAR String  
NAMEALT String  
DIFFASCII Integer  
NAMEASCII String  
ADM0CAP Double  
CAPALT Double  
CAPIN String  
WORLDCITY Double  
MEGACITY Integer  
SOV0NAME String  
SOV_A3 String  
ADM0NAME String  
ADM0_A3 String  
ADM1NAME String  
ISO_A2 String  
NOTE String  
LATITUDE Double  
LONGITUDE Double  
CHANGED Double  
NAMEDIFF Integer  
DIFFNOTE String  
POP_MAX Integer  
POP_MIN Integer  
POP_OTHER Integer  
GEONAMEID Double  
MEGANAME String  
LS_NAME String
```

LS_MATCH Integer
CHECKME Integer
MAX_POP10 Integer
MAX_POP20 Integer
MAX_POP50 Integer
MAX_POP300 Integer
MAX_POP310 Integer
MAX_NATSCA Integer
MIN_AREAKM Integer
MAX_AREAKM Double
MIN_AREAMI Double
MAX_AREAMI Double
MIN_PERKM Double
MAX_PERKM Double
MIN_PERMI Double
MAX_PERMI Double
MIN_BBXMINDouble
MAX_BBXMINDouble
MIN_BBXMINDouble
MAX_BBXMINDouble
MIN_BBYMINDouble
MAX_BBYMINDouble
MIN_BBYMINDouble
MAX_BBYMINDouble
MEAN_BBXC Double
MEAN_BBYC Double
COMPARE Integer
GN_ASCII String
FEATURE_CL String
FEATURE_CO String
ADMIN1_COD Double
GN_POP Integer
ELEVATION Double
GTOPO30 Double
TIMEZONE String
GEONAMESNO String
UN_FID Integer
UN_ADM0 String
UN_LAT Double
UN_LONG Double
POP1950 Double
POP1955 Double
POP1960 Double
POP1965 Double
POP1970 Double
POP1975 Double
POP1980 Double
POP1985 Double

POP1990 Double
 POP1995 Double
 POP2000 Double
 POP2005 Double
 POP2010 Double
 POP2015 Double
 POP2020 Double
 POP2025 Double
 POP2050 Double
 CITYALT String
 popDiff Integer
 popPerc Double
 ls_gross Integer
 X Double
 Y Double

```
geo-shell> layer features --name places_xy --filter "NAME='Seattle'" --field "NAME,X,Y"
```

```
Feature (fid—34f3c37f_17060e5e693_-6084)
```

```
-----  

NAME = Seattle  

X = -122.34193084586849  

Y = 47.57194791253073
```

Validity

Check for invalid geometries in the Layer.

```
geo-shell> layer validity --name areas
```

Name	Description	Mandatory	Specified Default	Unspecified Default
name	The Layer name	true		
fields	A comma delimited list of Fields to include	false		

```
geo-shell> workspace open --name areas --params src/test/resources/invalid.properties  

Workspace areas opened!
```

```
geo-shell> layer open --workspace areas --layer invalid --name areas  

Opened Workspace areas Layer invalid as areas
```

```
geo-shell> layer validity --name areas  

Values Reason
```

```
-----  

invalid.1360815594529 Self-intersection
```

Geoprocessing

Clip

Clip the input Layer by the other Layer to produce the output Layer

```
geo-shell> layer clip --input-name a --clip-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
clip-name	The clip Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params src/test/resources/layeralgebra.gpkg  
Workspace layers opened!
```

```
geo-shell> workspace open --name results --params memory  
Workspace results opened!
```

```
geo-shell> layer open --workspace layers --layer a --name a  
Opened Workspace layers Layer a as a
```

```
geo-shell> layer open --workspace layers --layer b --name b  
Opened Workspace layers Layer b as b
```

```
geo-shell> layer clip --input-name a --clip-name b --output-workspace results --output-name results  
Done clipping a to b to create results!
```

```
geo-shell> style vector default --layer a --color red --opacity 0.75 --file examples/red.sld  
Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!
```

```
geo-shell> style vector default --layer b --color green --opacity 0.75 --file examples/green.sld  
Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!
```

```
geo-shell> style vector default --layer results --color blue --opacity 0.75 --file examples/blue.sld  
Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!
```

```
geo-shell> layer style set --name a --style examples/red.sld  
Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a
```

```
geo-shell> layer style set --name b --style examples/green.sld  
Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b
```

```
geo-shell> layer style set --name results --style examples/blue.sld
```

Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

```
geo-shell> map open --name map  
Map map opened!
```

```
geo-shell> map add layer --name map --layer a  
Added a layer to map map
```

```
geo-shell> map add layer --name map --layer b  
Added b layer to map map
```

```
geo-shell> map add layer --name map --layer results  
Added results layer to map map
```

```
geo-shell> map draw --name map --file examples/layer_clip.png  
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_clip.png!
```

```
geo-shell> map close --name map  
Map map closed!
```



Convex Hull

Calculate the convexhull of the input Layer and save it to the output Layer.

```
geo-shell> layer convexhull --input-name countries --output-workspace layers --output-name  
convexhull
```

Name	Description	Mandatory	Specified Default	Unspecified Default
------	-------------	-----------	-------------------	---------------------

input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> layer convexhull --input-name countries --output-workspace layers --output-name
convexhull
Done!
```

```
geo-shell> style vector default --layer convexhull --color #1E90FF --opacity 0.25 --file
examples/convexhull.sld
Default Vector Style for convexhull written to /home/travis/build/jericks/geo-
shell/examples/convexhull.sld!
```

```
geo-shell> layer style set --name convexhull --style examples/convexhull.sld
Style /home/travis/build/jericks/geo-shell/examples/convexhull.sld set on convexhull
```

```
geo-shell> map open --name map
Map map opened!
```

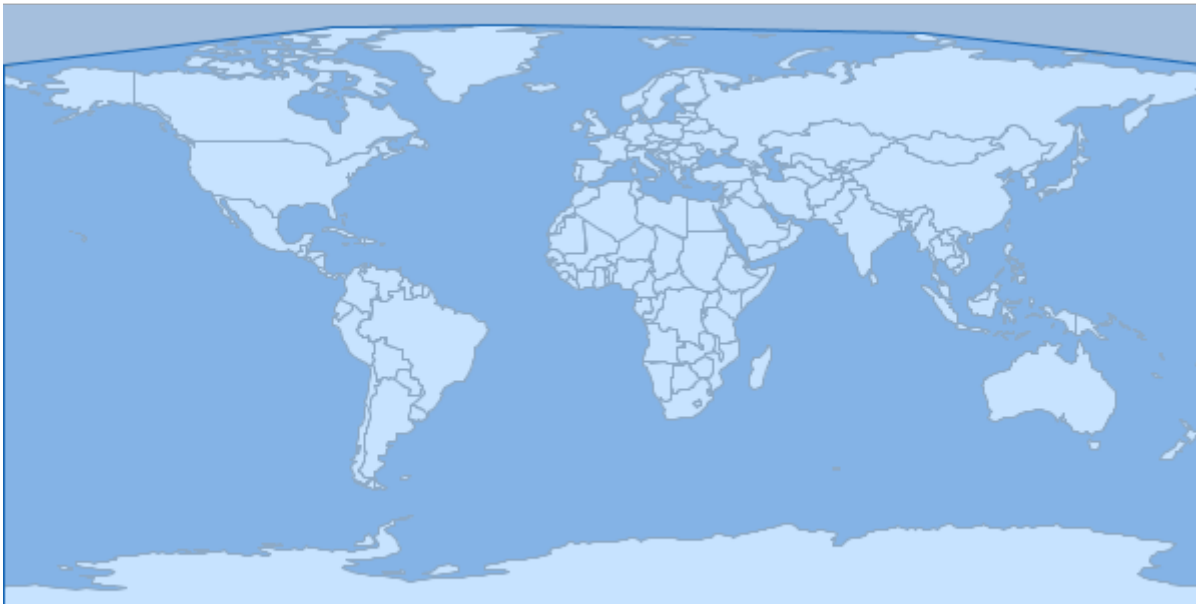
```
geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer convexhull
Added convexhull layer to map map
```

```
geo-shell> map draw --name map --file examples/layer_convexhull.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_convexhull.png!

geo-shell> map close --name map
Map map closed!
```



Convex Hulls

Calculate the convexhull of each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer convexhulls --input-name countries --output-workspace layers --output-name
convexhulls
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

geo-shell> **layer style set** --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **layer convexhulls** --input-name countries --output-workspace layers --output-name convexhulls
Done!

geo-shell> **style vector default** --layer convexhulls --color #1E90FF --opacity 0.25 --file examples/convexhulls.sld
Default Vector Style for convexhulls written to /home/travis/build/jericks/geo-shell/examples/convexhulls.sld!

geo-shell> **layer style set** --name convexhulls --style examples/convexhulls.sld
Style /home/travis/build/jericks/geo-shell/examples/convexhulls.sld set on convexhulls

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer convexhulls
Added convexhulls layer to map map

geo-shell> **map draw** --name map --file examples/layer_convexhulls.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_convexhulls.png!

geo-shell> **map close** --name map
Map map closed!



Coordinates

Extract the coordinates each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer coordinates --input-name states --output-workspace layers --output-name
coordinates
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer coordinates --input-name states --output-workspace layers --output-name
coordinates
Done!
```

```
geo-shell> style vector default --layer coordinates --color #1E90FF --opacity 0.75 --file
```

examples/coordinates.sld

Default Vector Style for coordinates written to /home/travis/build/jericks/geo-shell/examples/coordinates.sld!

geo-shell> **layer style set** --name coordinates --style examples/coordinates.sld

Style /home/travis/build/jericks/geo-shell/examples/coordinates.sld set on coordinates

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries

Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld

Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean

Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name map

Map map opened!

geo-shell> **map add layer** --name map --layer ocean

Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries

Added countries layer to map map

geo-shell> **map add layer** --name map --layer coordinates

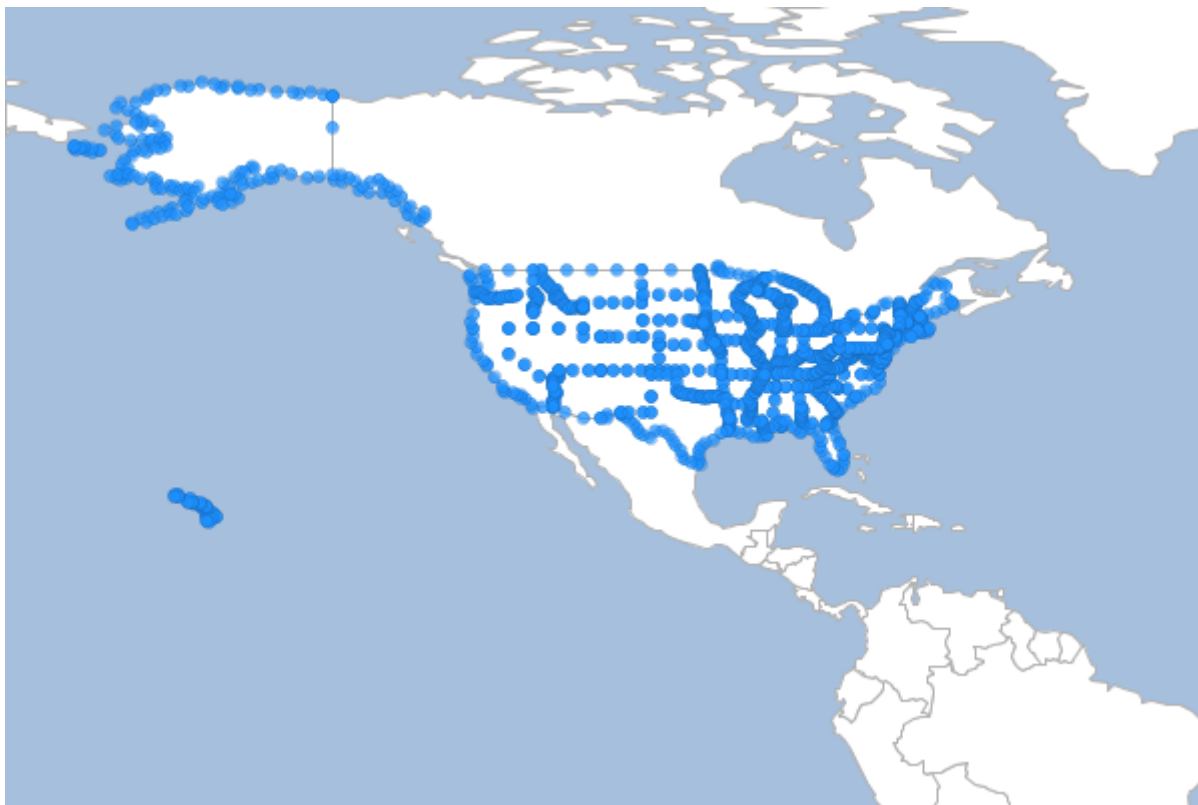
Added coordinates layer to map map

geo-shell> **map draw** --name map --file examples/layer_coordinates.png --bounds "-180,-8.233,-36.738,73.378"

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_coordinates.png!

geo-shell> **map close** --name map

Map map closed!



Delaunay

Calculate a delaunay diagram of the input Layer and save it to the output Layer.

```
geo-shell> layer delaunay --input-name places --output-workspace layers --output-name delaunay
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer places --name places
Opened Workspace naturalearth Layer places as places
```

```
geo-shell> layer delaunay --input-name places --output-workspace layers --output-name delaunay
Done!
```

geo-shell> **style vector default** --layer delaunay --color #1E90FF --opacity 0.25 --file examples/delaunay.sld
Default Vector Style for delaunay written to /home/travis/build/jericks/geo-shell/examples/delaunay.sld!

geo-shell> **layer style set** --name delaunay --style examples/delaunay.sld
Style /home/travis/build/jericks/geo-shell/examples/delaunay.sld set on delaunay

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name map
Map map opened!

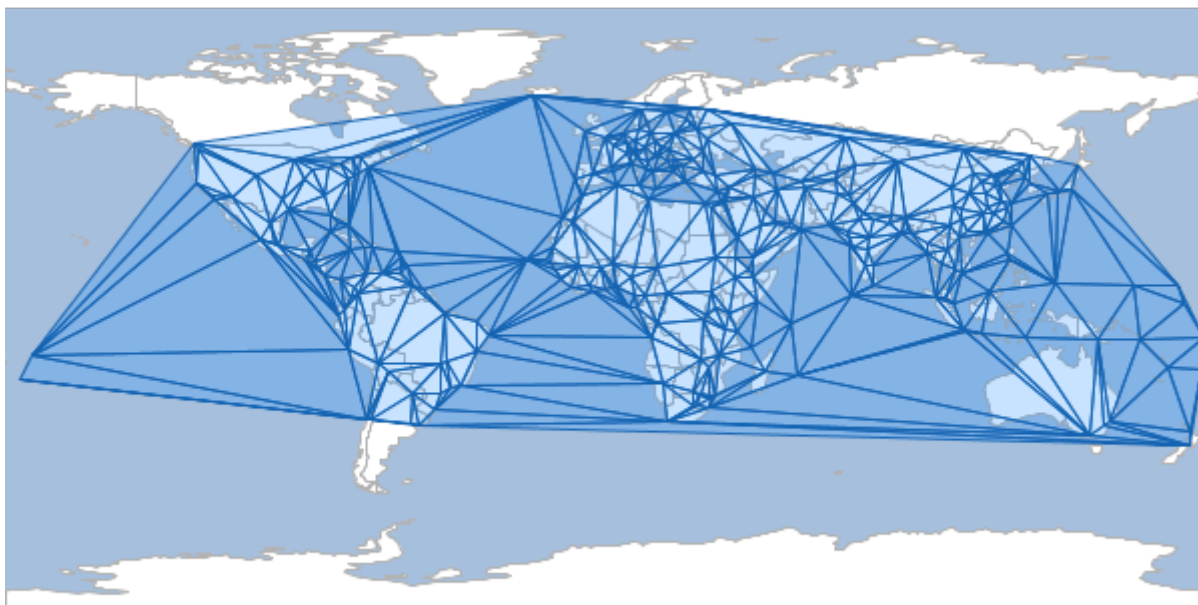
geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer delaunay
Added delaunay layer to map map

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_delaunay.png!
geo-shell> **map draw** --name map --file examples/layer_delaunay.png

Map map closed!
geo-shell> **map close** --name map



Densify

Densify the features of the input Layer and save them to the output Layer

```
geo-shell> layer densify --input-name states --output-workspace layers --output-name
states_densified --distance 0.1
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
distance	The distance tolerance	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer densify --input-name states --output-workspace layers --output-name
states_densified --distance 0.1
```


Done!

```
geo-shell> layer coordinates --input-name states_densified --output-workspace layers --output  
-name coordinates
```

Done!

```
geo-shell> style vector default --layer coordinates --color #1E90FF --opacity 0.75 --file  
examples/coordinates.sld
```

Default Vector Style for coordinates written to /home/travis/build/jericks/geo-shell/examples/coordinates.sld!

```
geo-shell> layer style set --name coordinates --style examples/coordinates.sld
```

Style /home/travis/build/jericks/geo-shell/examples/coordinates.sld set on coordinates

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
```

Opened Workspace naturalearth Layer countries as countries

```
geo-shell> layer style set --name countries --style examples/countries.sld
```

Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
```

Opened Workspace naturalearth Layer ocean as ocean

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
```

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

```
geo-shell> map open --name map
```

Map map opened!

```
geo-shell> map add layer --name map --layer ocean
```

Added ocean layer to map map

```
geo-shell> map add layer --name map --layer countries
```

Added countries layer to map map

Added coordinates layer to map map

```
geo-shell> map add layer --name map --layer coordinates
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_densify.png!

```
geo-shell> map draw --name map --file examples/layer_densify.png --bounds "-180,-8.233,-  
36.738,73.378"
```

Map map closed!

```
geo-shell> map close --name map
```



Dissolve

Dissolve the Features of a Layer by a Field.

```
geo-shell> layer dissolve --input-name states --output-workspace layers --output-name regions
--field SUB_REGION
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
field	The field to use to dissolve features	true		
idField	The name of the id field	false	id	id
countField	The name of the count field	false	count	count

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name shapefiles --params examples/states/states.shp
Workspace shapefiles opened!
```

geo-shell> **layer open** --workspace shapefiles --layer states --name states

Opened Workspace shapefiles Layer states as states

geo-shell> **layer dissolve** --input-name states --output-workspace layers --output-name regions
--field SUB_REGION

Done dissolving states to regions by SUB_REGION!

geo-shell> **style vector uniquevalues** --layer regions --field SUB_REGION --colors MutedTerrain
--file [silver] examples/regions.sld

Unique Values Vector Style for regions's SUB_REGION Field written to
/home/travis/build/jericks/geo-shell/examples/regions.sld!

geo-shell> **layer style set** --name regions --style examples/regions.sld

Style /home/travis/build/jericks/geo-shell/examples/regions.sld set on regions

geo-shell> **workspace open** --name naturalearth --params examples/naturalearth.gpkg

Workspace naturalearth opened!

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries

Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld

Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean

Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name map

Map map opened!

geo-shell> **map add layer** --name map --layer ocean

Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries

Added countries layer to map map

Added regions layer to map map

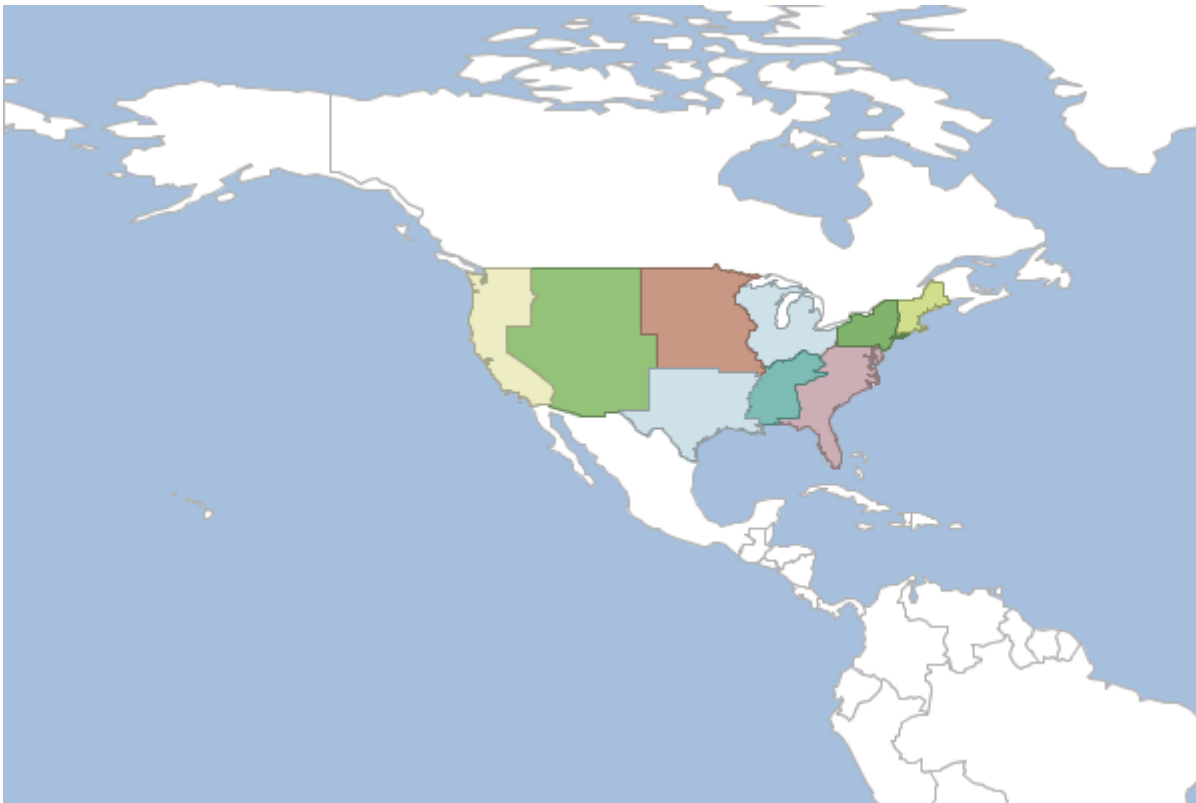
geo-shell> **map add layer** --name map --layer regions

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_dissolve.png!

geo-shell> **map draw** --name map --file examples/layer_dissolve.png --bounds "-180,-8.233,-
36.738,73.378"

Map map closed!

geo-shell> **map close** --name map



Erase

Erase one Layer from another Layer

```
geo-shell> layer erase --input-name a --other-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params src/test/resources/layeralgebra.gpkg
Workspace layers opened!
```

```
geo-shell> workspace open --name results --params memory
Workspace results opened!
```

```
geo-shell> layer open --workspace layers --layer a --name a
Opened Workspace layers Layer a as a
```

```
geo-shell> layer open --workspace layers --layer b --name b
Opened Workspace layers Layer b as b
```

geo-shell> **layer erase** --input-name a --other-name b --output-workspace results --output-name results

Done erasing a from b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld
Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld
Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld
Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld
Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld
Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld
Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map
Map map opened!

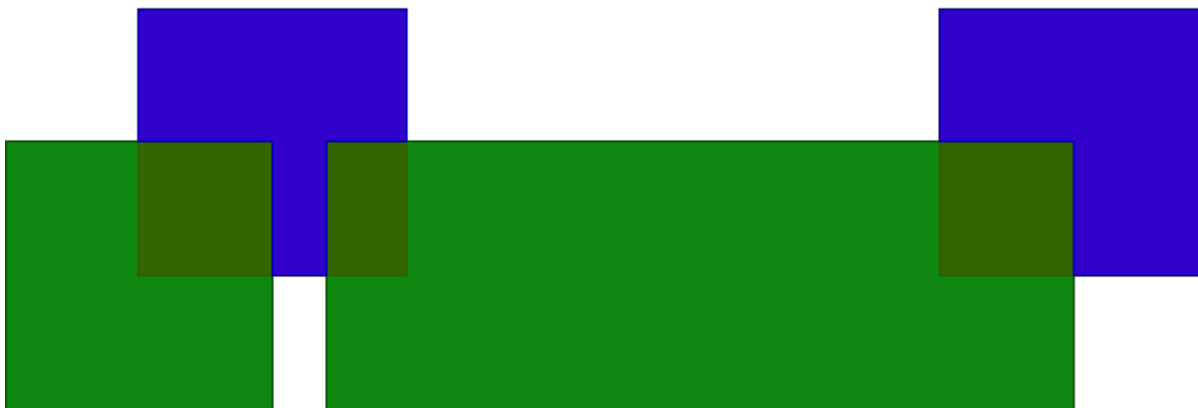
geo-shell> **map add layer** --name map --layer a
Added a layer to map map

geo-shell> **map add layer** --name map --layer b
Added b layer to map map

geo-shell> **map add layer** --name map --layer results
Added results layer to map map

geo-shell> **map draw** --name map --file examples/layer_erase.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_erase.png!

geo-shell> **map close** --name map
Map map closed!



Grid Row / Column

Create a grid Layer with rows and columns

```
geo-shell> layer grid rowcol --output-workspace layers --output-name rowcol --geometry -180,-90,180,90 --rows 10 --columns 8
```

Name	Description	Mandatory	Specified Default	Unspecified Default
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
rows	The number of rows	true		
columns	The number of columns	true		
geometry	The constraining geometry	true		
type	The geometry type (point or polygon)	false	polygon	polygon
projection	The projection	false	EPSG:4326	EPSG:4326
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer grid rowcol --output-workspace layers --output-name rowcol --geometry -180,-90,180,90 --rows 10 --columns 8
Done!

geo-shell> style vector default --layer rowcol --color #1E90FF --opacity 0.30 --file examples/rowcol.sld
Default Vector Style for rowcol written to /home/travis/build/jericks/geo-shell/examples/rowcol.sld!

geo-shell> layer style set --name rowcol --style examples/rowcol.sld
Style /home/travis/build/jericks/geo-shell/examples/rowcol.sld set on rowcol

geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!

geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> map open --name map
Map map opened!

geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map

geo-shell> map add layer --name map --layer countries
Added countries layer to map map

geo-shell> map add layer --name map --layer rowcol
Added rowcol layer to map map

geo-shell> map draw --name map --file examples/layer_grid_rowcol.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_grid_rowcol.png!

geo-shell> map close --name map
Map map closed!
```



Grid Width / Height

Create a grid Layer with cell width and height

```
geo-shell> layer grid widthheight --output-workspace layers --output-name widthheight --geometry
-180,-90,180,90 --cell-width 8 --cell-height 7
```

Name	Description	Mandatory	Specified Default	Unspecified Default
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
cell-width	The width of each cell	true		
cell-height	The height of each cell	true		
geometry	The constraining geometry	true		
type	The geometry type (point or polygon)	false	polygon	polygon
projection	The projection	false	EPSG:4326	EPSG:4326
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```



```
geo-shell> layer grid widthheight --output-workspace layers --output-name widthheight --geometry
-180,-90,180,90 --cell-width 8 --cell-height 7
Done!

geo-shell> style vector default --layer widthheight --color #1E90FF --opacity 0.30 --file
examples/widthheight.sld
Default Vector Style for widthheight written to /home/travis/build/jericks/geo-
shell/examples/widthheight.sld!

geo-shell> layer style set --name widthheight --style examples/widthheight.sld
Style /home/travis/build/jericks/geo-shell/examples/widthheight.sld set on widthheight

geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!

geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> map open --name map
Map map opened!

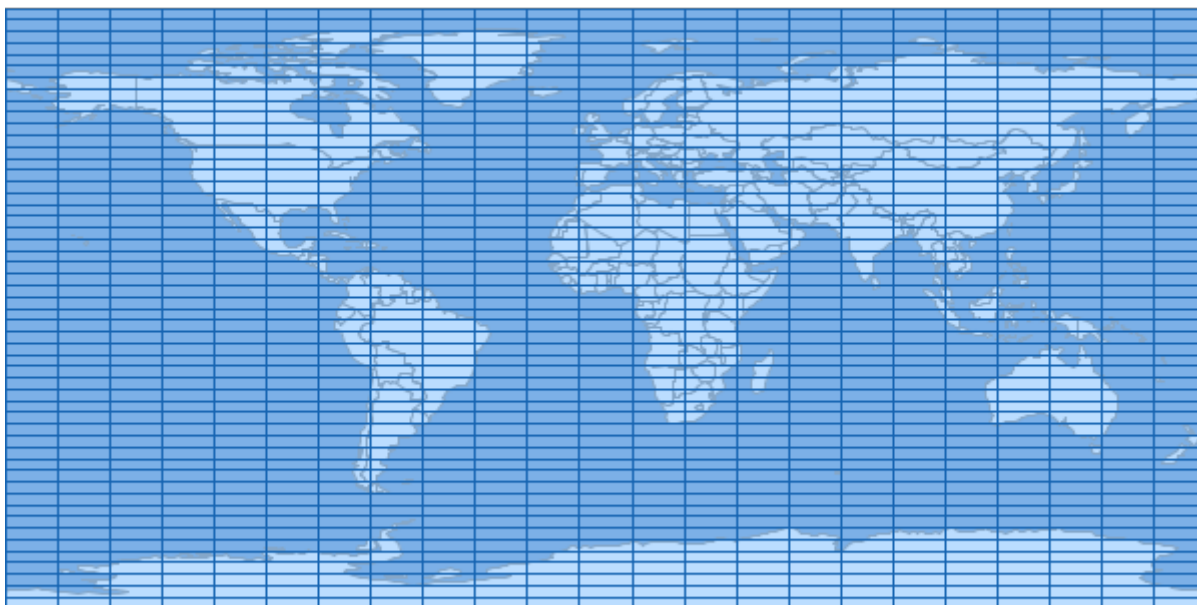
geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map

geo-shell> map add layer --name map --layer countries
Added countries layer to map map

geo-shell> map add layer --name map --layer widthheight
Added widthheight layer to map map

geo-shell> map draw --name map --file examples/layer_grid_widthheight.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_grid_widthheight.png!

geo-shell> map close --name map
Map map closed!
```



Identity

Calculate the intersection between a Layer with another Layer

```
geo-shell> layer identity --input-name a --other-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
postfix-all	Whether to postfix all field names when combining schemas	false	false	false
include-duplicates	Whether to include duplicate field names	false	true	true

```
geo-shell> workspace open --name layers --params src/test/resources/layeralgebra.gpkg
Workspace layers opened!
```

geo-shell> **workspace open** --name results --params memory
Workspace results opened!

geo-shell> **layer open** --workspace layers --layer a --name a
Opened Workspace layers Layer a as a

geo-shell> **layer open** --workspace layers --layer b --name b
Opened Workspace layers Layer b as b

geo-shell> **layer identity** --input-name a --other-name b --output-workspace results --output-name results
Done calculating the identity between a and b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld
Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld
Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld
Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld
Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld
Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld
Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map
Map map opened!

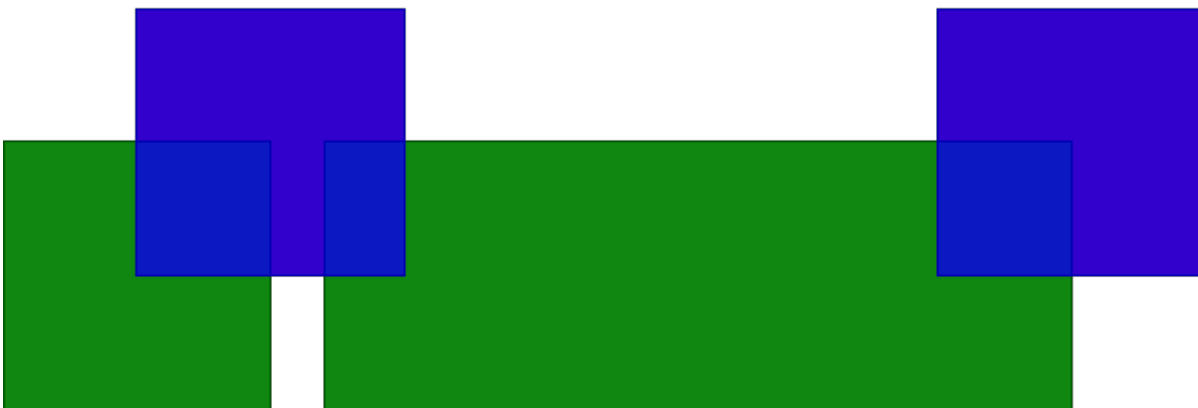
geo-shell> **map add layer** --name map --layer a
Added a layer to map map

geo-shell> **map add layer** --name map --layer b
Added b layer to map map

geo-shell> **map add layer** --name map --layer results
Added results layer to map map

geo-shell> **map draw** --name map --file examples/layer_identity.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_identity.png!

geo-shell> **map close** --name map
Map map closed!



Intersection

Calculate the intersection between a Layer with another Layer

geo-shell> **layer intersection** --input-name a --other-name b --output-workspace results --output-name results

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
postfix-all	Whether to postfix all field names when combining schemas	false	false	false
include-duplicates	Whether to include duplicate field names	false	true	true

geo-shell> **workspace open** --name layers --params src/test/resources/layeralgebra.gpkg
Workspace layers opened!

geo-shell> **workspace open** --name results --params memory

Workspace results opened!

geo-shell> **layer open** --workspace layers --layer a --name a

Opened Workspace layers Layer a as a

geo-shell> **layer open** --workspace layers --layer b --name b

Opened Workspace layers Layer b as b

geo-shell> **layer intersection** --input-name a --other-name b --output-workspace results --output -name results

Done calculating the intersection between a and b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld

Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld

Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld

Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld

Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld

Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld

Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map

Map map opened!

geo-shell> **map add layer** --name map --layer a

Added a layer to map map

geo-shell> **map add layer** --name map --layer b

Added b layer to map map

geo-shell> **map add layer** --name map --layer results

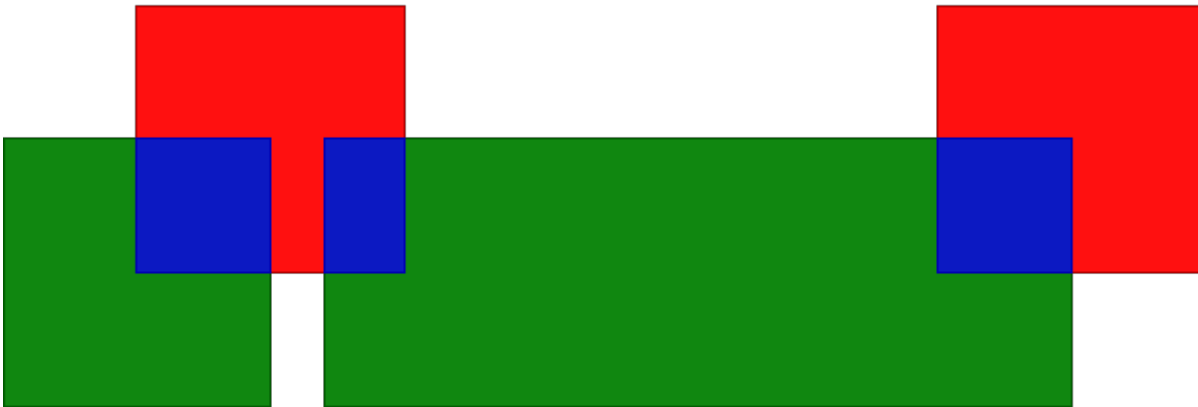
Added results layer to map map

geo-shell> **map draw** --name map --file examples/layer_intersection.png

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_intersection.png!

geo-shell> **map close** --name map

Map map closed!



Minimum Circle

Calculate the minimum bounding circle of the input Layer and save it to the output Layer.

```
geo-shell> layer mincircle --input-name countries --output-workspace layers --output-name mincircle
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **layer mincircle** --input-name countries --output-workspace layers --output-name mincircle
Done!

geo-shell> **style vector default** --layer mincircle --color #1E90FF --opacity 0.25 --file examples/mincircle.sld
Default Vector Style for mincircle written to /home/travis/build/jericks/geo-shell/examples/mincircle.sld!

geo-shell> **layer style set** --name mincircle --style examples/mincircle.sld
Style /home/travis/build/jericks/geo-shell/examples/mincircle.sld set on mincircle

geo-shell> **map open** --name map
Map map opened!

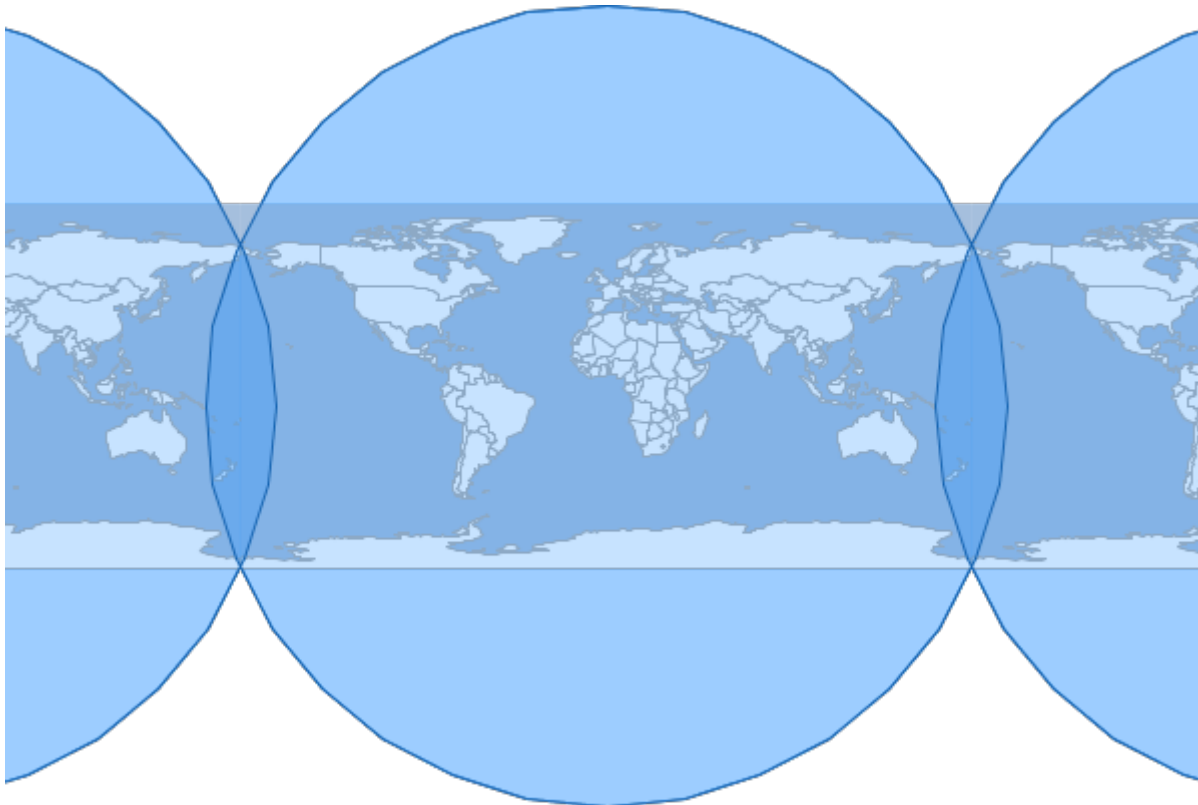
geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer mincircle
Added mincircle layer to map map

geo-shell> **map draw** --name map --file examples/layer_mincircle.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_mincircle.png!

geo-shell> **map close** --name map
Map map closed!



Minimum Circles

Calculate the minimum bounding circle of each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer mincircles --input-name countries --output-workspace layers --output-name mincircles
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
```


Opened Workspace naturalearth Layer ocean as ocean

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
```

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

```
geo-shell> layer mincircles --input-name countries --output-workspace layers --output-name mincircles
```

Done!

```
geo-shell> style vector default --layer mincircles --color #1E90FF --opacity 0.25 --file examples/mincircles.sld
```

Default Vector Style for mincircles written to /home/travis/build/jericks/geo-shell/examples/mincircles.sld!

```
geo-shell> layer style set --name mincircles --style examples/mincircles.sld
```

Style /home/travis/build/jericks/geo-shell/examples/mincircles.sld set on mincircles

```
geo-shell> map open --name map
```

Map map opened!

```
geo-shell> map add layer --name map --layer ocean
```

Added ocean layer to map map

```
geo-shell> map add layer --name map --layer countries
```

Added countries layer to map map

```
geo-shell> map add layer --name map --layer mincircles
```

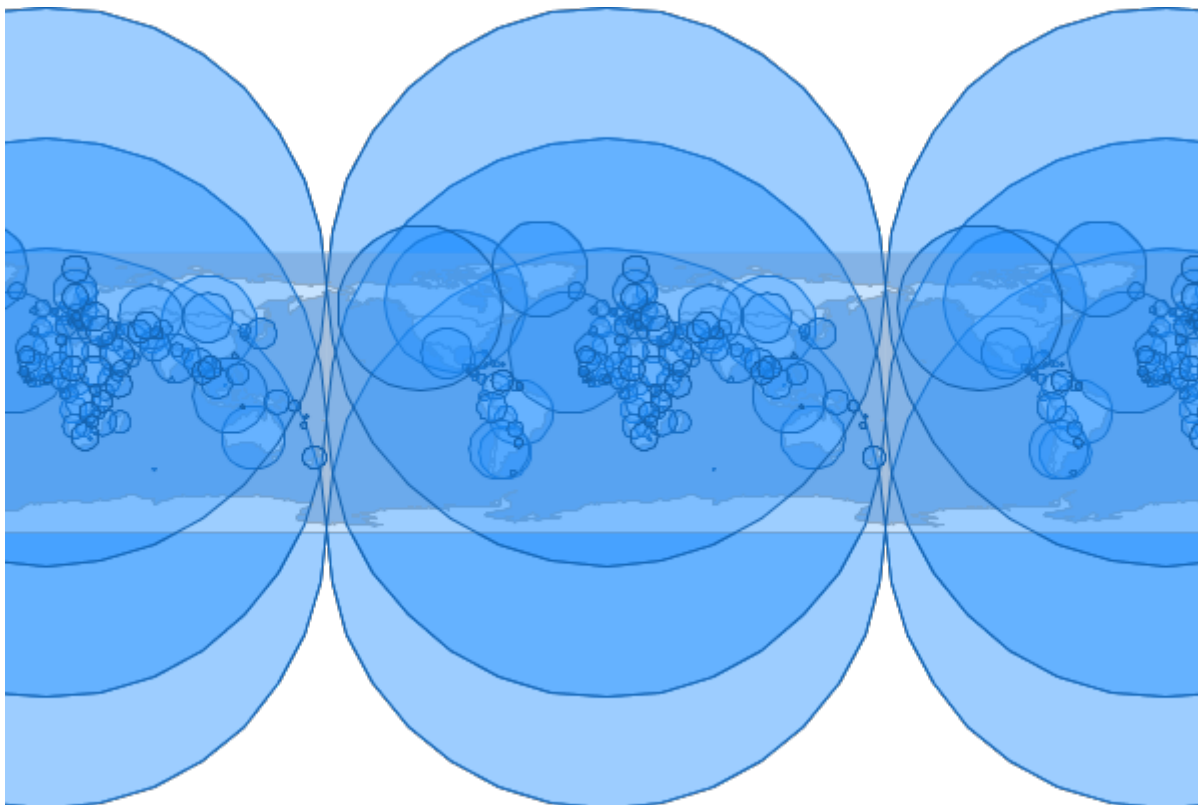
Added mincircles layer to map map

```
geo-shell> map draw --name map --file examples/layer_mincircles.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_mincircles.png!

```
geo-shell> map close --name map
```

Map map closed!



Minimum Rectangle

Calculate the minimum rectangle of the input Layer and save it to the output Layer.

```
geo-shell> layer minrect --input-name countries --output-workspace layers --output-name minrect
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **layer minrect** --input-name countries --output-workspace layers --output-name minrect
Done!

geo-shell> **style vector default** --layer minrect --color #1E90FF --opacity 0.25 --file
examples/minrect.sld
Default Vector Style for minrect written to /home/travis/build/jericks/geo-shell/examples/minrect.sld!

geo-shell> **layer style set** --name minrect --style examples/minrect.sld
Style /home/travis/build/jericks/geo-shell/examples/minrect.sld set on minrect

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer minrect
Added minrect layer to map map

geo-shell> **map draw** --name map --file examples/layer_minrect.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_minrect.png!

geo-shell> **map close** --name map
Map map closed!



Minimum Rectangles

Calculate the minimum rectangle of each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer minrects --input-name countries --output-workspace layers --output-name minrects
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
```

Opened Workspace naturalearth Layer ocean as ocean

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
```

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

```
geo-shell> layer minrects --input-name countries --output-workspace layers --output-name minrects
```

Done!

```
geo-shell> style vector default --layer minrects --color #1E90FF --opacity 0.25 --file examples/minrects.sld
```

Default Vector Style for minrects written to /home/travis/build/jericks/geo-shell/examples/minrects.sld!

```
geo-shell> layer style set --name minrects --style examples/minrects.sld
```

Style /home/travis/build/jericks/geo-shell/examples/minrects.sld set on minrects

```
geo-shell> map open --name map
```

Map map opened!

```
geo-shell> map add layer --name map --layer ocean
```

Added ocean layer to map map

```
geo-shell> map add layer --name map --layer countries
```

Added countries layer to map map

```
geo-shell> map add layer --name map --layer minrects
```

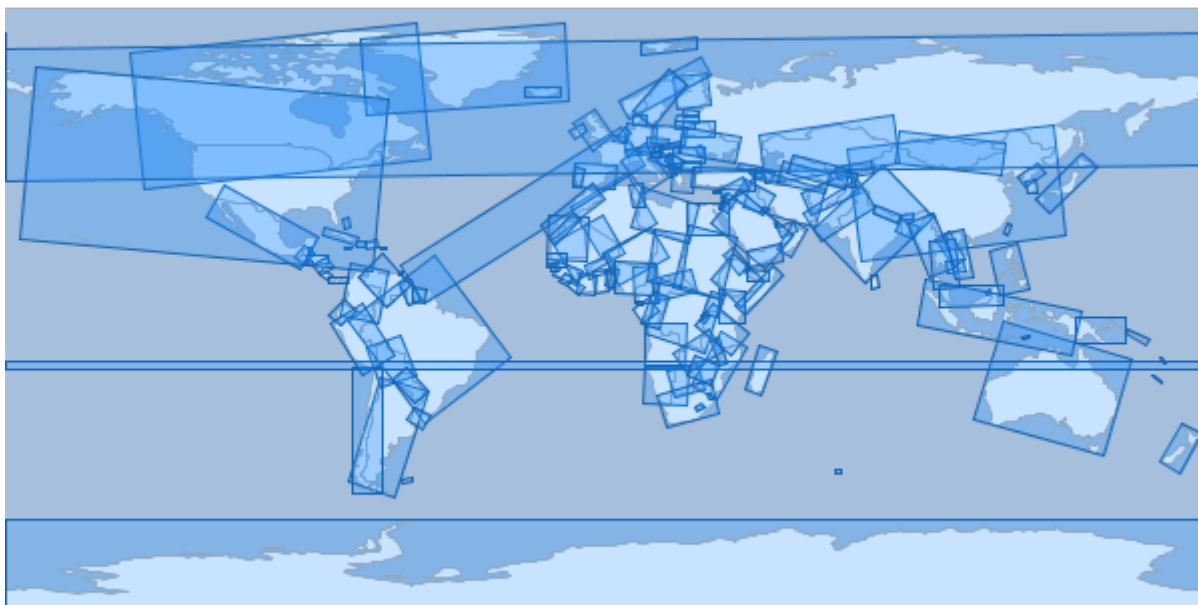
Added minrects layer to map map

```
geo-shell> map draw --name map --file examples/layer_minrects.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_minrects.png!

```
geo-shell> map close --name map
```

Map map closed!



Octangle Envelope

Calculate the octagonal envelope of the input Layer and save it to the output Layer.

```
geo-shell> layer octagonalenvelope --input-name countries --output-workspace layers --output-name octagonalenvelope
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **layer octagonalenvelope** --input-name countries --output-workspace layers --output
-name octagonalenvelope
Done!

geo-shell> **style vector default** --layer octagonalenvelope --color #1E90FF --opacity 0.25 --file
examples/octagonalenvelope.sld
Default Vector Style for octagonalenvelope written to /home/travis/build/jericks/geo-
shell/examples/octagonalenvelope.sld!

geo-shell> **layer style set** --name octagonalenvelope --style examples/octagonalenvelope.sld
Style /home/travis/build/jericks/geo-shell/examples/octagonalenvelope.sld set on octagonalenvelope

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer octagonalenvelope
Added octagonalenvelope layer to map map

geo-shell> **map draw** --name map --file examples/layer_octagonalenvelope.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_octagonalenvelope.png!

geo-shell> **map close** --name map
Map map closed!



Octangle Envelopes

Calculate the octagonal envelope of each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer octagonalenvelopes --input-name countries --output-workspace layers --output
-name octagonalenvelopes
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
```


Opened Workspace naturalearth Layer ocean as ocean

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
```

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

```
geo-shell> layer octagonalenvelopes --input-name countries --output-workspace layers --output  
-name octagonalenvelopes
```

Done!

```
geo-shell> style vector default --layer octagonalenvelopes --color #1E90FF --opacity 0.25 --file  
examples/octagonalenvelopes.sld
```

Default Vector Style for octagonalenvelopes written to /home/travis/build/jericks/geo-shell/examples/octagonalenvelopes.sld!

```
geo-shell> layer style set --name octagonalenvelopes --style examples/octagonalenvelopes.sld
```

Style /home/travis/build/jericks/geo-shell/examples/octagonalenvelopes.sld set on octagonalenvelopes

```
geo-shell> map open --name map
```

Map map opened!

```
geo-shell> map add layer --name map --layer ocean
```

Added ocean layer to map map

```
geo-shell> map add layer --name map --layer countries
```

Added countries layer to map map

```
geo-shell> map add layer --name map --layer octagonalenvelopes
```

Added octagonalenvelopes layer to map map

```
geo-shell> map draw --name map --file examples/layer_octagonalenvelopes.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_octagonalenvelopes.png!

```
geo-shell> map close --name map
```

Map map closed!



Points Along Lines

Create points along lines

```
geo-shell> layer points along lines --input-name mississippi --output-workspace layers --output-name points --distance 2.0
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
distance	The distance between points	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name rivers --params
src/test/resources/rivers/ne_110m_rivers_lake_centerlines.shp
Workspace rivers opened!
```

```
geo-shell> layer open --workspace rivers --layer ne_110m_rivers_lake_centerlines --name rivers
Opened Workspace rivers Layer ne_110m_rivers_lake_centerlines as rivers
```

```
geo-shell> layer copy --input-name rivers --output-workspace layers --output-name mississippi #
```

[gray]--filter# "name='Mississippi'"

Done!

geo-shell> **style vector default** --layer mississippi --color blue --file examples/river.sld
Default Vector Style for mississippi written to /home/travis/build/jericks/geo-shell/examples/river.sld!

geo-shell> **layer style set** --name mississippi --style examples/river.sld
Style /home/travis/build/jericks/geo-shell/examples/river.sld set on mississippi

geo-shell> **layer points along lines** --input-name mississippi --output-workspace layers --output -name points --distance 2.0
Done placing points along mississippi every 2.0 to create points!

geo-shell> **style vector default** --layer points --color green --file examples/points.sld
Default Vector Style for points written to /home/travis/build/jericks/geo-shell/examples/points.sld!

geo-shell> **layer style set** --name points --style examples/points.sld
Style /home/travis/build/jericks/geo-shell/examples/points.sld set on points

geo-shell> **workspace open** --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer mississippi
Added mississippi layer to map map

geo-shell> **map add layer** --name map --layer points
Added points layer to map map

geo-shell> **map draw** --name map --file examples/layer_points_along_lines.png --bounds "-180,-

8.233,-36.738,73.378"

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_points_along_lines.png!



Simplify

Simplify the features of the input Layer and save them to the output Layer

```
geo-shell> layer simplify --input-name mississippi --output-workspace layers --output-name simplified --distance 1.0
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
algorithm	The simplify algorithm (DouglasPeucker - dp or TopologyPreserving - tp)	false	tp	tp
distance	The distance tolerance	true		

```
geo-shell> workspace open --name layers --params memory  
Workspace layers opened!
```

```
geo-shell> workspace open --name rivers --params  
src/test/resources/rivers/ne_110m_rivers_lake_centerlines.shp  
Workspace rivers opened!
```

```
geo-shell> layer open --workspace rivers --layer ne_110m_rivers_lake_centerlines --name rivers  
Opened Workspace rivers Layer ne_110m_rivers_lake_centerlines as rivers
```

```
geo-shell> layer copy --input-name rivers --output-workspace layers --output-name mississippi #  
[gray]--filter# "name='Mississippi'"  
Done!
```

```
geo-shell> layer simplify --input-name mississippi --output-workspace layers --output-name  
simplified --distance 1.0  
Done!
```

```
geo-shell> style vector default --layer simplified --color blue --file examples/river.sld  
Default Vector Style for simplified written to /home/travis/build/jericks/geo-shell/examples/river.sld!
```

```
geo-shell> layer style set --name simplified --style examples/river.sld  
Style /home/travis/build/jericks/geo-shell/examples/river.sld set on simplified
```

```
geo-shell> layer coordinates --input-name simplified --output-workspace layers --output-name  
points  
Done!
```

```
geo-shell> style vector default --layer points --color green --file examples/points.sld  
Default Vector Style for points written to /home/travis/build/jericks/geo-shell/examples/points.sld!
```

```
geo-shell> layer style set --name points --style examples/points.sld  
Style /home/travis/build/jericks/geo-shell/examples/points.sld set on points
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld  
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld  
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map  
Map map opened!
```

```
geo-shell> map add layer --name map --layer ocean  
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries
```

Added countries layer to map map

```
geo-shell> map add layer --name map --layer simplified
```

Added simplified layer to map map

```
geo-shell> map add layer --name map --layer points
```

Added points layer to map map

```
geo-shell> map draw --name map --file examples/layer_simplify.png --bounds "-180,-8.233,-36.738,73.378"
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_simplify.png!

```
geo-shell> map close --name map
```

Map map closed!



Symmetric Difference

Calculate the symmetric difference between a Layer and another Layer.

```
geo-shell> layer symdifference --input-name a --other-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		

output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
postfix-all	Whether to postfix all field names when combining schemas	false	false	false
include-duplicates	Whether to include duplicate field names	false	true	true

geo-shell> **workspace open** --name layers --params src/test/resources/layeralgebra.gpkg
Workspace layers opened!

geo-shell> **workspace open** --name results --params memory
Workspace results opened!

geo-shell> **layer open** --workspace layers --layer a --name a
Opened Workspace layers Layer a as a

geo-shell> **layer open** --workspace layers --layer b --name b
Opened Workspace layers Layer b as b

geo-shell> **layer symdifference** --input-name a --other-name b --output-workspace results --output-name results
Done calculating the symmetric difference between a and b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld
Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld
Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld
Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld
Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld
Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld
Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer a

Added a layer to map map

```
geo-shell> map add layer --name map --layer b
```

Added b layer to map map

```
geo-shell> map add layer --name map --layer results
```

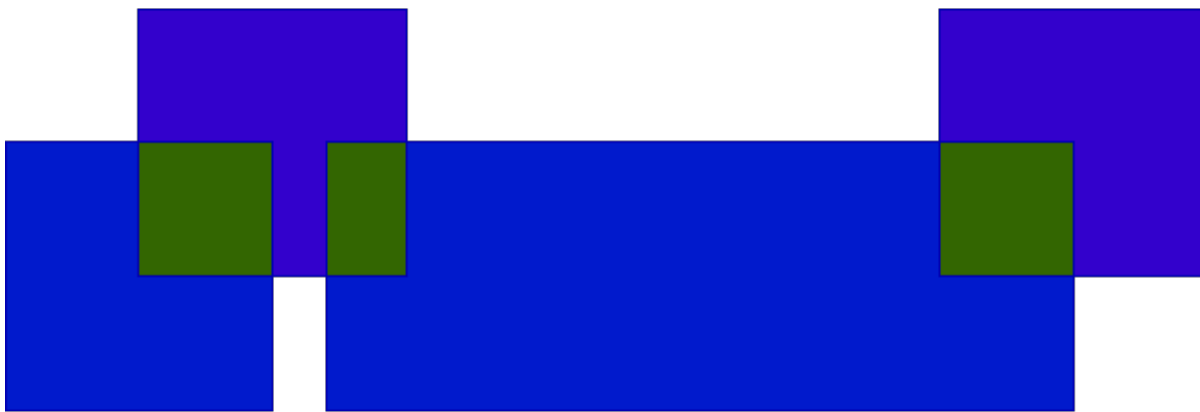
Added results layer to map map

```
geo-shell> map draw --name map --file examples/layer_symdifference.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_symdifference.png!

```
geo-shell> map close --name map
```

Map map closed!



Transform

Transform the features of the input Layer and save them to the output Layer

```
geo-shell> layer transform --input-name points --output-workspace layers --output-name polys  
--transforms "the_geom=buffer(the_geom, 5)|id=id*10"
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

transforms	The pipe delimited list of transforms (field=expression or function)	true		
------------	--	------	--	--

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer random --output-workspace layers --output-name points --geometry -180,-90,180,90
--number 100 --projection EPSG:4326
Done!
```

```
geo-shell> style vector default --layer points --color #1E90FF --file examples/points.sld
Default Vector Style for points written to /home/travis/build/jericks/geo-shell/examples/points.sld!
```

```
geo-shell> layer style set --name points --style examples/points.sld
Style /home/travis/build/jericks/geo-shell/examples/points.sld set on points
```

```
geo-shell> layer transform --input-name points --output-workspace layers --output-name polys
--transforms "the_geom=buffer(the_geom, 5)|id=id*10"
Done transforming points to polys with the_geom=buffer(the_geom, 5)|id=id*10!
```

```
geo-shell> style vector default --layer polys --color blue --opacity 0.25 --file examples/polys.sld
Default Vector Style for polys written to /home/travis/build/jericks/geo-shell/examples/polys.sld!
```

```
geo-shell> layer style set --name polys --style examples/polys.sld
Style /home/travis/build/jericks/geo-shell/examples/polys.sld set on polys
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map
Map map opened!
```

```
geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer polys
```

Added polys layer to map map

```
geo-shell> map add layer --name map --layer points
```

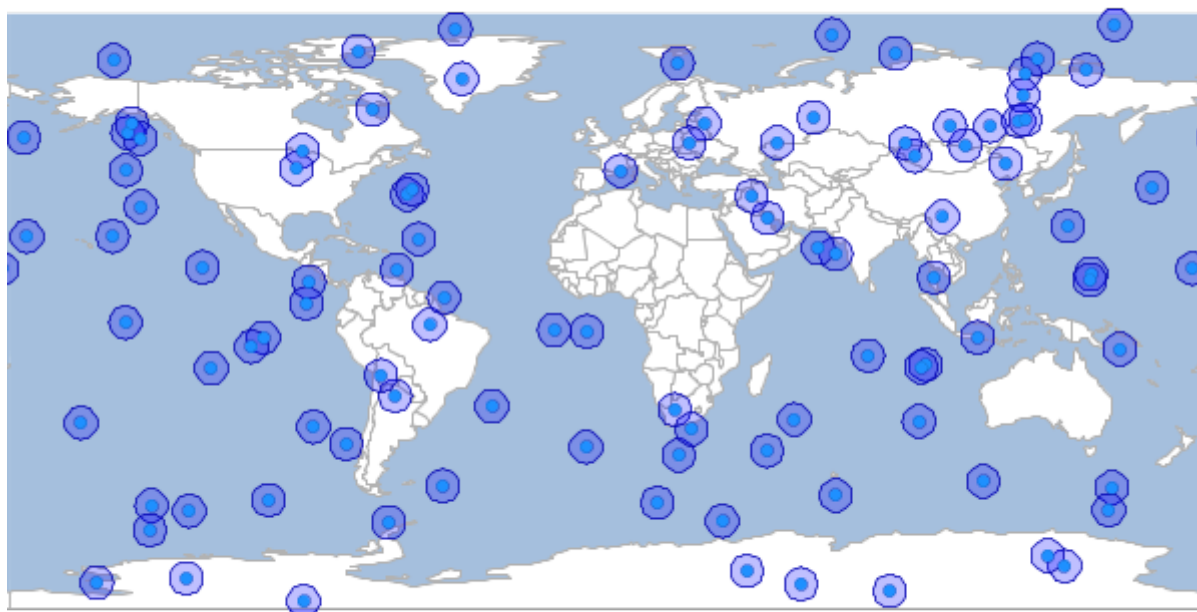
Added points layer to map map

```
geo-shell> map draw --name map --file examples/layer_transform.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_transform.png!

```
geo-shell> map close --name map
```

Map map closed!



Union

Union a Layer with another Layer

```
geo-shell> layer union --input-name a --other-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

postfix-all	Whether to postfix all field names when combining schemas	false	false	false
include-duplicates	Whether to include duplicate field names	false	true	true

geo-shell> **workspace open** --name layers --params src/test/resources/layeralgebra.gpkg
Workspace layers opened!

geo-shell> **workspace open** --name results --params memory
Workspace results opened!

geo-shell> **layer open** --workspace layers --layer a --name a
Opened Workspace layers Layer a as a

geo-shell> **layer open** --workspace layers --layer b --name b
Opened Workspace layers Layer b as b

geo-shell> **layer union** --input-name a --other-name b --output-workspace results --output-name results
Done unioning a and b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld
Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld
Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld
Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld
Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld
Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld
Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer a
Added a layer to map map

geo-shell> **map add layer** --name map --layer b
Added b layer to map map

```
geo-shell> map add layer --name map --layer results
```

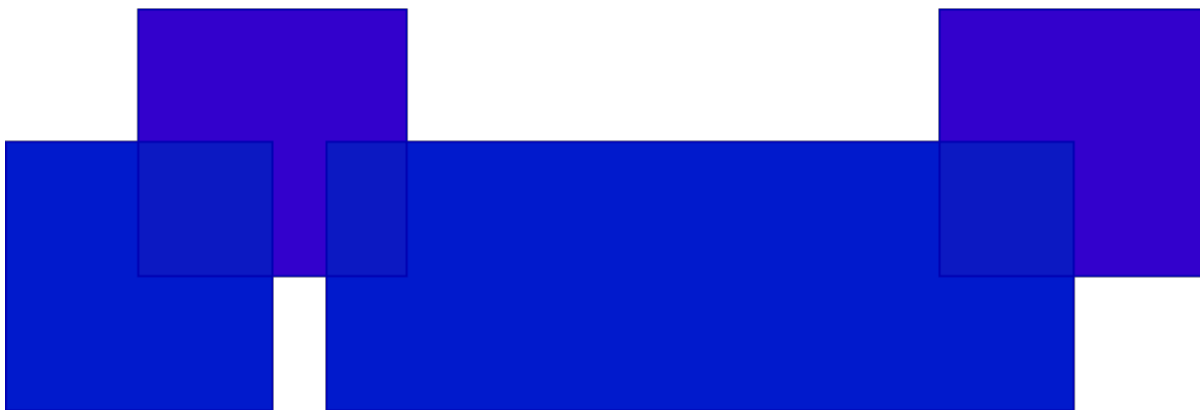
Added results layer to map map

```
geo-shell> map draw --name map --file examples/layer_union.png
```

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_union.png!

```
geo-shell> map close --name map
```

Map map closed!



Update

Calculate the update between a Layer with another Layer

```
geo-shell> layer update --input-name a --other-name b --output-workspace results --output-name results
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
other-name	The other Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params src/test/resources/layeralgebra.gpkg
```

Workspace layers opened!

geo-shell> **workspace open** --name results --params memory

Workspace results opened!

geo-shell> **layer open** --workspace layers --layer a --name a

Opened Workspace layers Layer a as a

geo-shell> **layer open** --workspace layers --layer b --name b

Opened Workspace layers Layer b as b

geo-shell> **layer update** --input-name a --other-name b --output-workspace results --output-name results

Done calculating the update between a and b to create results!

geo-shell> **style vector default** --layer a --color red --opacity 0.75 --file examples/red.sld

Default Vector Style for a written to /home/travis/build/jericks/geo-shell/examples/red.sld!

geo-shell> **style vector default** --layer b --color green --opacity 0.75 --file examples/green.sld

Default Vector Style for b written to /home/travis/build/jericks/geo-shell/examples/green.sld!

geo-shell> **style vector default** --layer results --color blue --opacity 0.75 --file examples/blue.sld

Default Vector Style for results written to /home/travis/build/jericks/geo-shell/examples/blue.sld!

geo-shell> **layer style set** --name a --style examples/red.sld

Style /home/travis/build/jericks/geo-shell/examples/red.sld set on a

geo-shell> **layer style set** --name b --style examples/green.sld

Style /home/travis/build/jericks/geo-shell/examples/green.sld set on b

geo-shell> **layer style set** --name results --style examples/blue.sld

Style /home/travis/build/jericks/geo-shell/examples/blue.sld set on results

geo-shell> **map open** --name map

Map map opened!

geo-shell> **map add layer** --name map --layer a

Added a layer to map map

geo-shell> **map add layer** --name map --layer b

Added b layer to map map

geo-shell> **map add layer** --name map --layer results

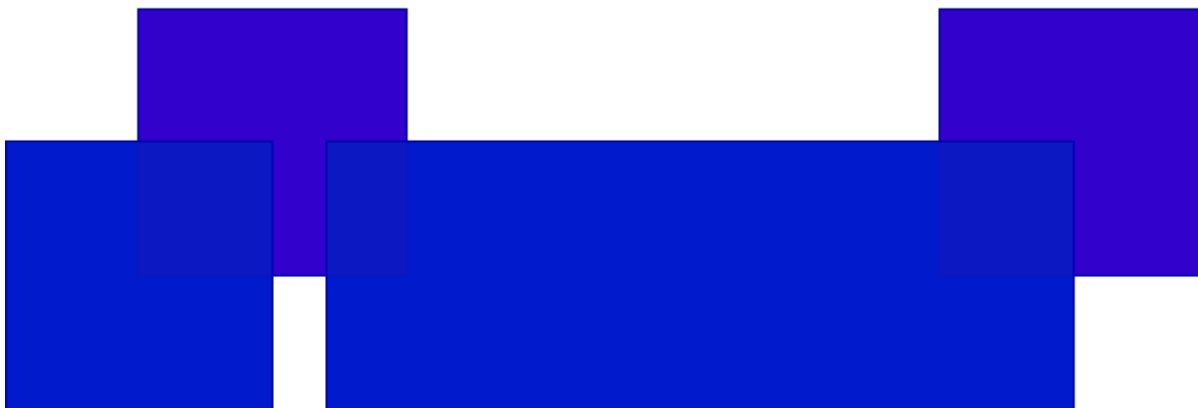
Added results layer to map map

geo-shell> **map draw** --name map --file examples/layer_update.png

Done drawing /home/travis/build/jericks/geo-shell/examples/layer_update.png!

geo-shell> **map close** --name map

Map map closed!



Voronoi

Calculate a voronoi diagram of the input Layer and save it to the output Layer.

```
geo-shell> layer voronoi --input-name places --output-workspace layers --output-name voronoi
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer places --name places
Opened Workspace naturalearth Layer places as places
```

```
geo-shell> layer voronoi --input-name places --output-workspace layers --output-name voronoi
Done!
```

```
geo-shell> style vector default --layer voronoi --color #1E90FF --opacity 0.25 --file
examples/voronoi.sld
Default Vector Style for voronoi written to /home/travis/build/jericks/geo-
shell/examples/voronoi.sld!
```

```
geo-shell> layer style set --name voronoi --style examples/voronoi.sld
Style /home/travis/build/jericks/geo-shell/examples/voronoi.sld set on voronoi
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map
Map map opened!
```

```
geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer voronoi
Added voronoi layer to map map
```

```
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_voronoi.png!
geo-shell> map draw --name map --file examples/layer_voronoi.png --bounds -180,-90,180,90
```

```
Map map closed!
geo-shell> map close --name map
```



Random Points

Create a Layer with a number of randomly located points

```
geo-shell> layer random --output-workspace layers --output-name points --geometry -180,-90,180,90
--number 100 --projection EPSG:4326
```

Name	Description	Mandatory	Specified Default	Unspecified Default
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
number	The number of points	true		
geometry	The geometry or bounds in which to create the points	true		
projection	The projection	true		
id-field	The id field name	false	id	id
geometry-field	The geometry field name	false	the_geom	the_geom
grid	Whether to create points in a grid	false	false	false

constrained-to-circle	Whether points should be constrained to a circle	false	false	false
gutter-fraction	The size of gutter between cells	false	0	0

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer random --output-workspace layers --output-name points --geometry -180,-90,180,90
--number 100 --projection EPSG:4326
Done!
```

```
geo-shell> style vector default --layer points --color #1E90FF --file examples/points.sld
Default Vector Style for points written to /home/travis/build/jericks/geo-shell/examples/points.sld!
```

```
geo-shell> layer style set --name points --style examples/points.sld
Style /home/travis/build/jericks/geo-shell/examples/points.sld set on points
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name randomMap
Map randomMap opened!
```

```
geo-shell> map add layer --name randomMap --layer ocean
Added ocean layer to map randomMap
```

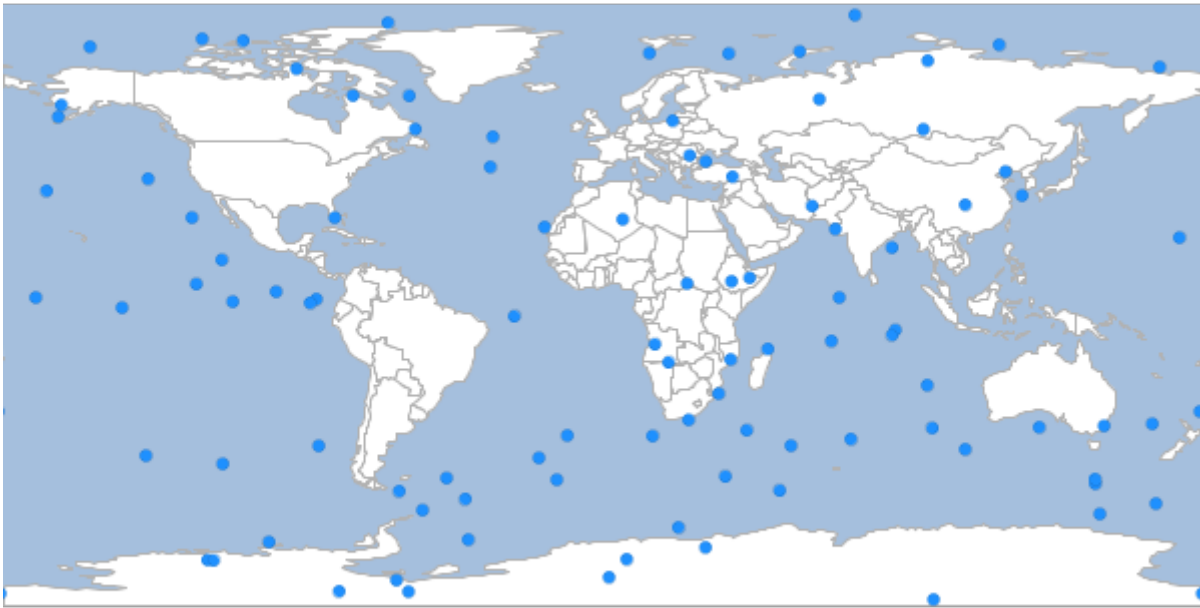
```
geo-shell> map add layer --name randomMap --layer countries
Added countries layer to map randomMap
```

```
geo-shell> map add layer --name randomMap --layer points
Added points layer to map randomMap
```

```
geo-shell> map draw --name randomMap --file examples/random_points.png
Done drawing /home/travis/build/jericks/geo-shell/examples/random_points.png!
```

```
geo-shell> map close --name randomMap
```

Map randomMap closed!



Buffer

Buffer the input Layer to the output Layer.

```
geo-shell> layer buffer --input-name points --output-workspace layers --output-name buffers  
--distance 10
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
distance	The buffer distance	true		

```
geo-shell> workspace open --name layers --params memory  
Workspace layers opened!
```

```
geo-shell> layer random --output-workspace layers --output-name points --geometry -180,-90,180,90  
--number 100 --projection EPSG:4326  
Done!
```

```
geo-shell> layer buffer --input-name points --output-workspace layers --output-name buffers  
--distance 10
```

Done!

```
geo-shell> style vector default --layer points --color #1E90FF --file examples/points.sld  
Default Vector Style for points written to /home/travis/build/jericks/geo-shell/examples/points.sld!
```

```
geo-shell> style vector default --layer buffers --color #1E90FF --opacity 0.25 --file  
examples/buffers.sld  
Default Vector Style for buffers written to /home/travis/build/jericks/geo-shell/examples/buffers.sld!
```

```
geo-shell> layer style set --name points --style examples/points.sld  
Style /home/travis/build/jericks/geo-shell/examples/points.sld set on points
```

```
geo-shell> layer style set --name buffers --style examples/buffers.sld  
Style /home/travis/build/jericks/geo-shell/examples/buffers.sld set on buffers
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld  
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld  
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map  
Map map opened!
```

```
geo-shell> map add layer --name map --layer ocean  
Added ocean layer to map map
```

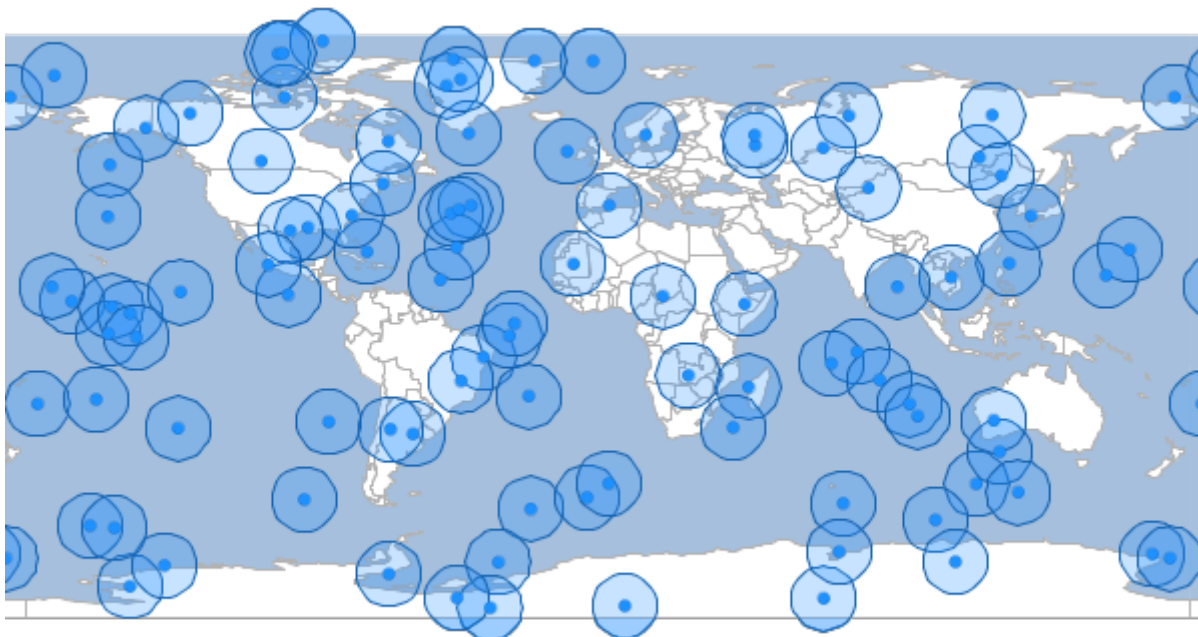
```
geo-shell> map add layer --name map --layer countries  
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer buffers  
Added buffers layer to map map
```

```
geo-shell> map add layer --name map --layer points  
Added points layer to map map
```

```
geo-shell> map draw --name map --file examples/layer_buffer.png  
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_buffer.png!
```

```
geo-shell> map close --name map  
Map map closed!
```



Centroid

Calculate the centroids of the input Layer to the output Layer.

```
geo-shell> layer centroid --input-name countries --output-name centroids --output-workspace layers
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer centroid --input-name countries --output-name centroids --output-workspace layers
```

Done!

```
geo-shell> style vector default --layer centroids --color #1E90FF --file examples/centroids.sld  
Default Vector Style for centroids written to /home/travis/build/jericks/geo-shell/examples/centroids.sld!
```

```
geo-shell> layer style set --name centroids --style examples/centroids.sld  
Style /home/travis/build/jericks/geo-shell/examples/centroids.sld set on centroids
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld  
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map  
Map map opened!
```

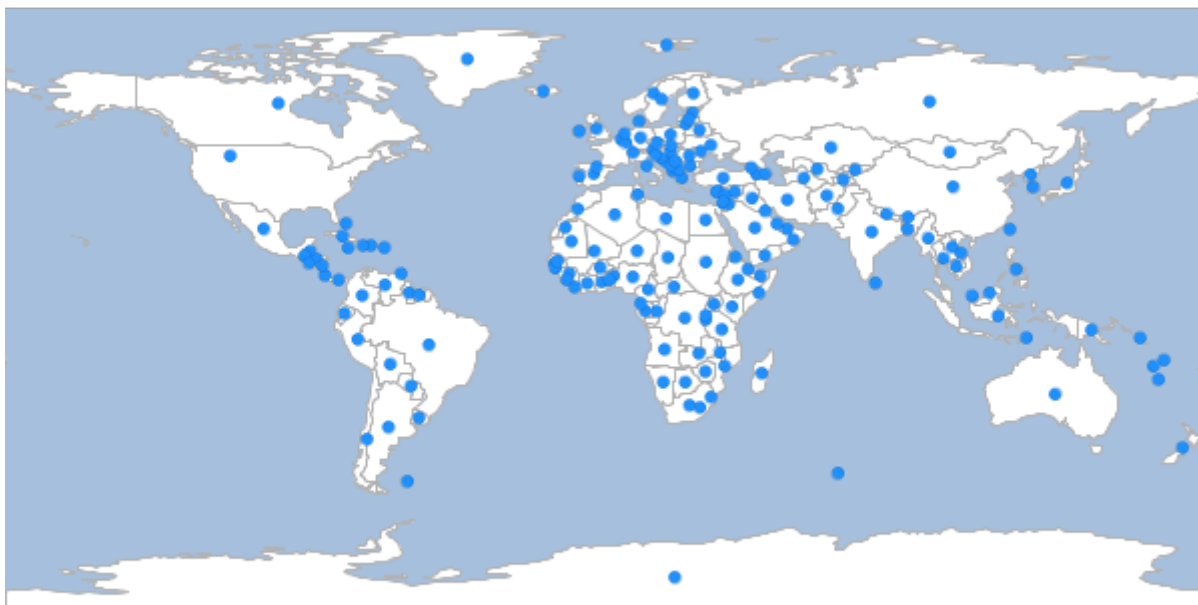
```
geo-shell> map add layer --name map --layer ocean  
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries  
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer centroids  
Added centroids layer to map map
```

```
geo-shell> map draw --name map --file examples/layer_centroid.png  
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_centroid.png!
```

```
geo-shell> map close --name map  
Map map closed!
```



Interior Point

Calculate the interior points of the input Layer to the output Layer.

```
geo-shell> layer interiorpoint --input-name countries --output-name interiorpoints --output
-workspace layers
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer interiorpoint --input-name countries --output-name interiorpoints --output
-workspace layers
```

Done!

```
geo-shell> style vector default --layer interiorpoints --color #1E90FF --file
examples/interiorpoints.sld
```

Default Vector Style for interiorpoints written to /home/travis/build/jericks/geo-shell/examples/interiorpoints.sld!

```
geo-shell> layer style set --name interiorpoints --style examples/interiorpoints.sld
Style /home/travis/build/jericks/geo-shell/examples/interiorpoints.sld set on interiorpoints
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name map
Map map opened!
```

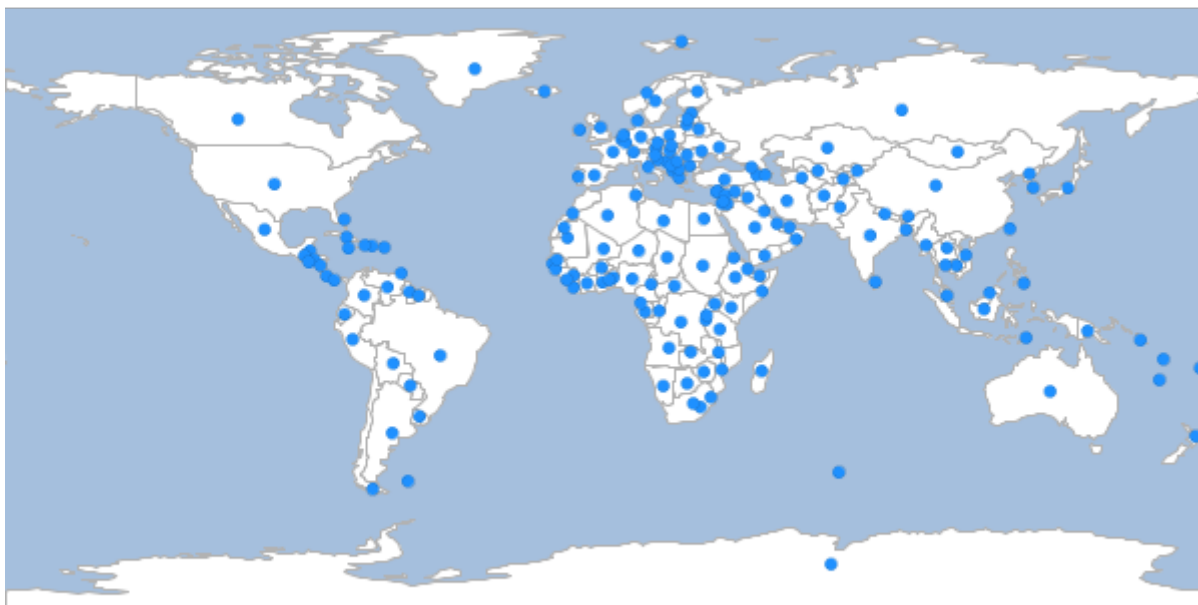
```
geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map
```

```
geo-shell> map add layer --name map --layer countries
Added countries layer to map map
```

```
geo-shell> map add layer --name map --layer interiorpoints
Added interiorpoints layer to map map
```

```
geo-shell> map draw --name map --file examples/layer_interiorpoint.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_interiorpoint.png!
```

```
geo-shell> map close --name map
Map map closed!
```



Extent

Calculate the extent of the input Layer and save it to the output Layer.

```
geo-shell> layer extent --input-name states --output-workspace layers --output-name usa
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		
geometry-field	The geometry field name	false	the_geom	the_geom

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer style set --name states --style examples/states.sld
Unable to find Layer states
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```


geo-shell> **layer extent** --input-name states --output-workspace layers --output-name usa
Done!

geo-shell> **style vector default** --layer usa --color #1E90FF --opacity 0.25 --file examples/extent.sld
Default Vector Style for usa written to /home/travis/build/jericks/geo-shell/examples/extent.sld!

geo-shell> **layer style set** --name usa --style examples/extent.sld
Style /home/travis/build/jericks/geo-shell/examples/extent.sld set on usa

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name map
Map map opened!

geo-shell> **map add layer** --name map --layer ocean
Added ocean layer to map map

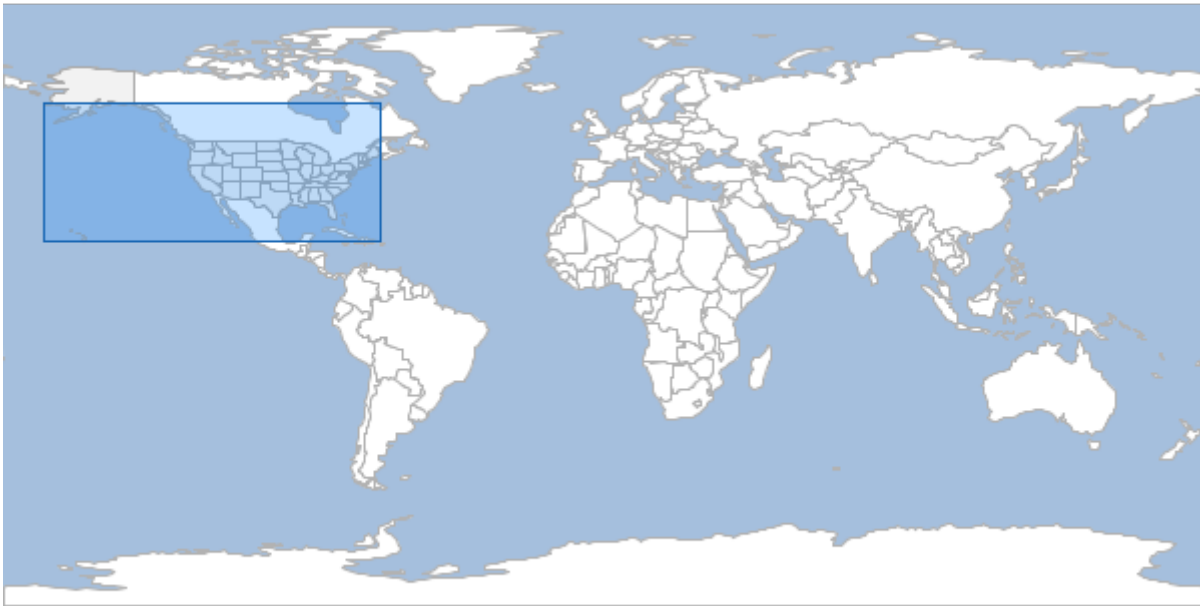
geo-shell> **map add layer** --name map --layer countries
Added countries layer to map map

geo-shell> **map add layer** --name map --layer states
Added states layer to map map

geo-shell> **map add layer** --name map --layer usa
Added usa layer to map map

geo-shell> **map draw** --name map --file examples/layer_extent.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_extent.png!

geo-shell> **map close** --name map
Map map closed!



Extents

Calculate the extents of each Feature in the input Layer and save them to the output Layer.

```
geo-shell> layer extents --input-name states --output-workspace layers --output-name state_extents
```

Name	Description	Mandatory	Specified Default	Unspecified Default
input-name	The Layer name	true		
output-workspace	The output Layer Workspace	true		
output-name	The output Layer name	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
Workspace naturalearth opened!
```

```
geo-shell> layer style set --name states --style examples/states.sld
Unable to find Layer states
```

```
geo-shell> layer open --workspace naturalearth --layer states --name states
Opened Workspace naturalearth Layer states as states
```

```
geo-shell> layer extents --input-name states --output-workspace layers --output-name state_extents
Done!
```

```
geo-shell> style vector default --layer state_extents --color #1E90FF --opacity 0.25 --file
examples/extent.sld
Default Vector Style for state_extents written to /home/travis/build/jericks/geo-
shell/examples/extent.sld!

geo-shell> layer style set --name state_extents --style examples/extent.sld
Style /home/travis/build/jericks/geo-shell/examples/extent.sld set on state_extents

geo-shell> layer open --workspace naturalearth --layer countries --name countries
Opened Workspace naturalearth Layer countries as countries

geo-shell> layer style set --name countries --style examples/countries.sld
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> layer open --workspace naturalearth --layer ocean --name ocean
Opened Workspace naturalearth Layer ocean as ocean

geo-shell> layer style set --name ocean --style examples/ocean.sld
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> map open --name map
Map map opened!

geo-shell> map add layer --name map --layer ocean
Added ocean layer to map map

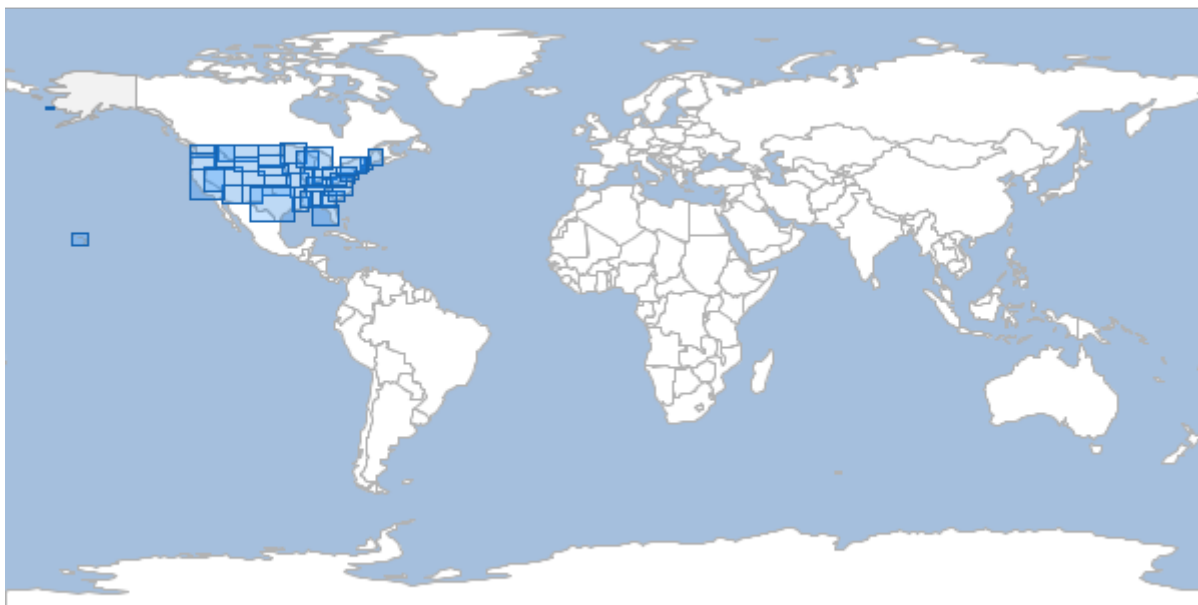
geo-shell> map add layer --name map --layer countries
Added countries layer to map map

geo-shell> map add layer --name map --layer states
Added states layer to map map

geo-shell> map add layer --name map --layer state_extents
Added state_extents layer to map map

geo-shell> map draw --name map --file examples/layer_extents.png
Done drawing /home/travis/build/jericks/geo-shell/examples/layer_extents.png!

geo-shell> map close --name map
Map map closed!
```



Graticule

Square

Create a square graticule.

```
geo-shell> layer graticule square --workspace layers --name squares --bounds -180,-90,180,90
--length 20
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
name	The new Layer name	true		
bounds	The bounds	true		
length	The length	true		
spacing	The spacing	false	-1	-1

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer graticule square --workspace layers --name squares --bounds -180,-90,180,90
--length 20
Created Square Graticule Layer squares!
```

```
geo-shell> style vector default --layer squares --color #1E90FF --opacity 0.30 --file
```

examples/squares.sld

Default Vector Style for squares written to /home/travis/build/jericks/geo-shell/examples/squares.sld!

geo-shell> **layer style set** --name squares --style examples/squares.sld

Style /home/travis/build/jericks/geo-shell/examples/squares.sld set on squares

geo-shell> **workspace open** --name naturalearth --params examples/naturalearth.gpkg

Workspace naturalearth opened!

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries

Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld

Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean

Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name graticule

Map graticule opened!

geo-shell> **map add layer** --name graticule --layer ocean

Added ocean layer to map graticule

geo-shell> **map add layer** --name graticule --layer countries

Added countries layer to map graticule

geo-shell> **map add layer** --name graticule --layer squares

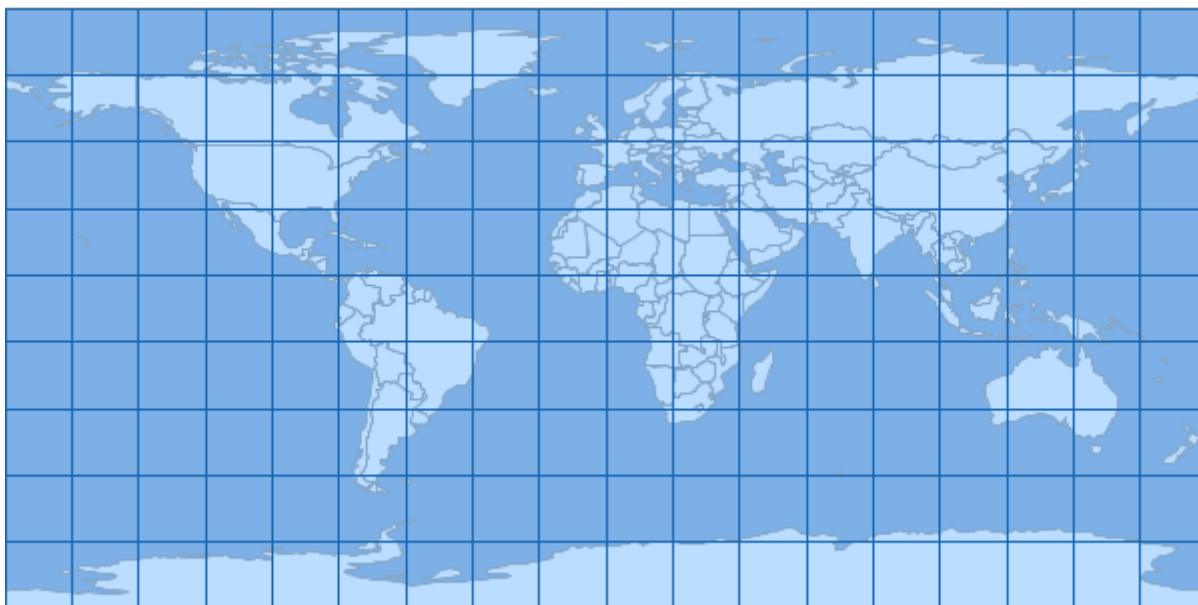
Added squares layer to map graticule

geo-shell> **map draw** --name graticule --file examples/square_graticules.png

Done drawing /home/travis/build/jericks/geo-shell/examples/square_graticules.png!

geo-shell> **map close** --name graticule

Map graticule closed!



Rectangle

Create a rectangle graticule.

```
geo-shell> layer graticule rectangle --workspace layers --name rectangles --bounds -180,-90,180,90
--width 20 --height 10
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
name	The new Layer name	true		
bounds	The bounds	true		
width	The width	true		
height	The height	true		
spacing	The spacing	false	-1	-1

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer graticule rectangle --workspace layers --name rectangles --bounds -180,-90,180,90
--width 20 --height 10
Created Rectangle Graticule Layer rectangles!
```

```
geo-shell> style vector default --layer rectangles --color #1E90FF --opacity 0.30 --file
examples/rectangles.sld
```

Default Vector Style for rectangles written to /home/travis/build/jericks/geo-shell/examples/rectangles.sld!

```
geo-shell> layer style set --name rectangles --style examples/rectangles.sld  
Style /home/travis/build/jericks/geo-shell/examples/rectangles.sld set on rectangles
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg  
Workspace naturalearth opened!
```

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld  
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld  
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name graticule  
Map graticule opened!
```

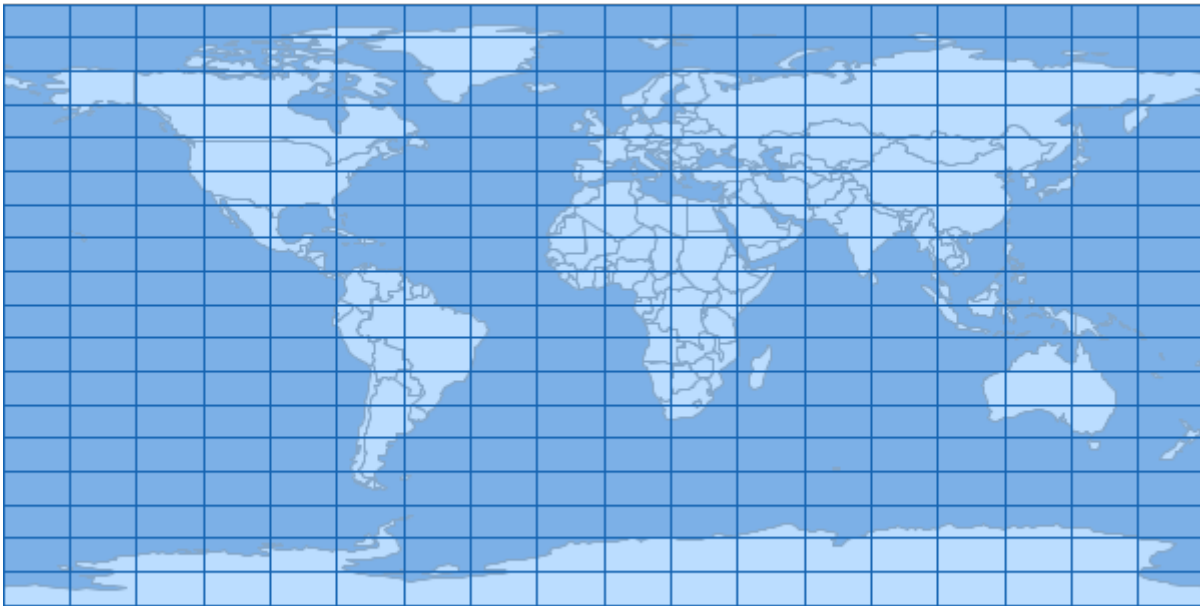
```
geo-shell> map add layer --name graticule --layer ocean  
Added ocean layer to map graticule
```

```
geo-shell> map add layer --name graticule --layer countries  
Added countries layer to map graticule
```

```
geo-shell> map add layer --name graticule --layer rectangles  
Added rectangles layer to map graticule
```

```
geo-shell> map draw --name graticule --file examples/rectangle_graticules.png  
Done drawing /home/travis/build/jericks/geo-shell/examples/rectangle_graticules.png!
```

```
geo-shell> map close --name graticule  
Map graticule closed!
```



Oval

Create a oval graticule.

```
geo-shell> layer graticule oval --workspace layers --name ovals --bounds -180,-90,180,90 --size 20
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
name	The new Layer name	true		
bounds	The bounds	true		
size	The size	true		

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer graticule oval --workspace layers --name ovals --bounds -180,-90,180,90 --size 20
Created Oval Graticule Layer ovals!
```

```
geo-shell> style vector default --layer ovals --color #1E90FF --opacity 0.30 --file examples/ovals.sld
Default Vector Style for ovals written to /home/travis/build/jericks/geo-shell/examples/ovals.sld!
```

```
geo-shell> layer style set --name ovals --style examples/ovals.sld
Style /home/travis/build/jericks/geo-shell/examples/ovals.sld set on ovals
```

```
geo-shell> workspace open --name naturalearth --params examples/naturalearth.gpkg
```


Workspace naturalearth opened!

```
geo-shell> layer open --workspace naturalearth --layer countries --name countries  
Opened Workspace naturalearth Layer countries as countries
```

```
geo-shell> layer style set --name countries --style examples/countries.sld  
Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries
```

```
geo-shell> layer open --workspace naturalearth --layer ocean --name ocean  
Opened Workspace naturalearth Layer ocean as ocean
```

```
geo-shell> layer style set --name ocean --style examples/ocean.sld  
Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean
```

```
geo-shell> map open --name graticule  
Map graticule opened!
```

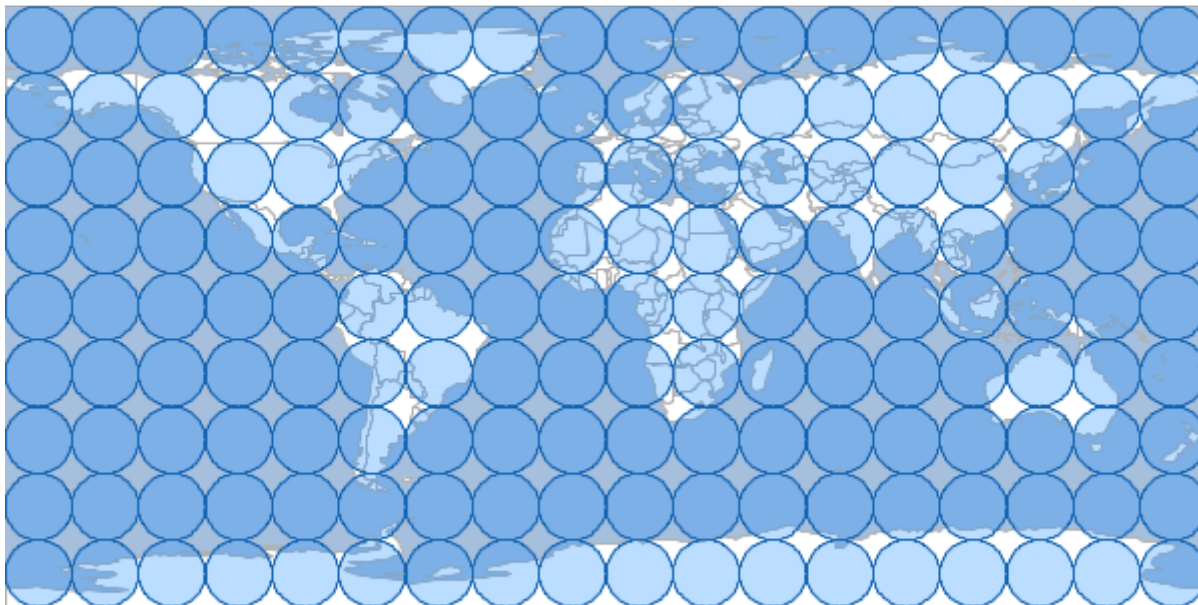
```
geo-shell> map add layer --name graticule --layer ocean  
Added ocean layer to map graticule
```

```
geo-shell> map add layer --name graticule --layer countries  
Added countries layer to map graticule
```

```
geo-shell> map add layer --name graticule --layer ovals  
Added ovals layer to map graticule
```

```
geo-shell> map draw --name graticule --file examples/oval_graticules.png  
Done drawing /home/travis/build/jericks/geo-shell/examples/oval_graticules.png!
```

```
geo-shell> map close --name graticule  
Map graticule closed!
```



Hexagon

Create a hexagon graticule.

```
geo-shell> layer graticule hexagon --workspace layers --name hexagons --bounds -180,-90,180,90
--length 10
```

Name	Description	Mandatory	Specified Default	Unspecified Default
workspace	The Workspace name	true		
name	The new Layer name	true		
bounds	The bounds	true		
length	The length	true		
spacing	The spacing	false	5	5
orientation	The orientation (flat or angled)	false	flat	flat

```
geo-shell> workspace open --name layers --params memory
Workspace layers opened!
```

```
geo-shell> layer graticule hexagon --workspace layers --name hexagons --bounds -180,-90,180,90
--length 10
Created Hexagon Graticule Layer hexagons!
```

```
geo-shell> style vector default --layer hexagons --color #1E90FF --opacity 0.30 --file
```

examples/hexagons.sld

Default Vector Style for hexagons written to /home/travis/build/jericks/geo-shell/examples/hexagons.sld!

geo-shell> **layer style set** --name hexagons --style examples/hexagons.sld

Style /home/travis/build/jericks/geo-shell/examples/hexagons.sld set on hexagons

geo-shell> **workspace open** --name naturalearth --params examples/naturalearth.gpkg

Workspace naturalearth opened!

geo-shell> **layer open** --workspace naturalearth --layer countries --name countries

Opened Workspace naturalearth Layer countries as countries

geo-shell> **layer style set** --name countries --style examples/countries.sld

Style /home/travis/build/jericks/geo-shell/examples/countries.sld set on countries

geo-shell> **layer open** --workspace naturalearth --layer ocean --name ocean

Opened Workspace naturalearth Layer ocean as ocean

geo-shell> **layer style set** --name ocean --style examples/ocean.sld

Style /home/travis/build/jericks/geo-shell/examples/ocean.sld set on ocean

geo-shell> **map open** --name graticule

Map graticule opened!

geo-shell> **map add layer** --name graticule --layer ocean

Added ocean layer to map graticule

geo-shell> **map add layer** --name graticule --layer countries

Added countries layer to map graticule

geo-shell> **map add layer** --name graticule --layer hexagons

Added hexagons layer to map graticule

geo-shell> **map draw** --name graticule --file examples/hexagon_graticules.png

Done drawing /home/travis/build/jericks/geo-shell/examples/hexagon_graticules.png!

geo-shell> **map close** --name graticule

Map graticule closed!

