Table of Contents

R	ender Recipes	. 1
	Creating Maps	. 1
	Rendering Maps	. 1
	Displaying Maps	17
	Drawing	19
	Plotting	27

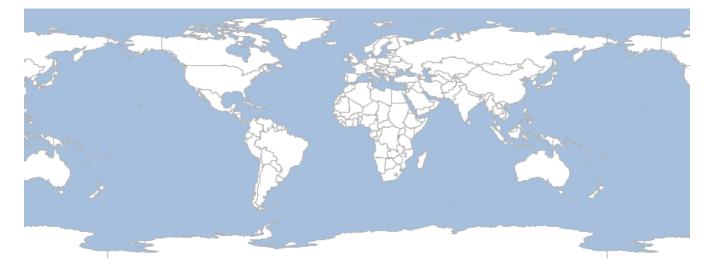
Render Recipes

The Render classes are in the **geoscript.render** package.

Creating Maps

Create a Map with Layers

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
File file = new File("map.png")
map.render(file)
```



Rendering Maps

Finding Renderers

Get all Renderers

```
List<Renderer> renderers = Renderers.list()
renderers.each { Renderer renderer ->
    println renderer.class.simpleName
}
```

```
ASCII
Base64
GeoTIFF
GIF
JPEG
Pdf
PNG
Svg
```

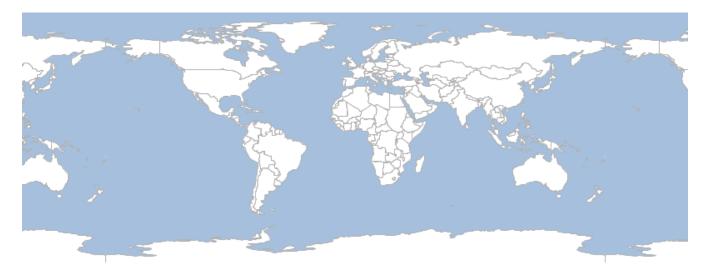
Get a Renderer

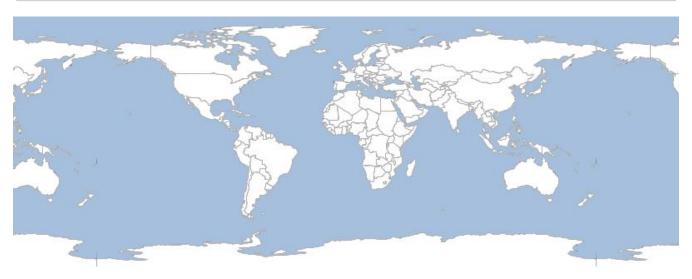
```
Renderer renderer = Renderers.find("png")
println renderer.class.simpleName
```

PNG

Image

Render a Map to an image using an Image Renderer

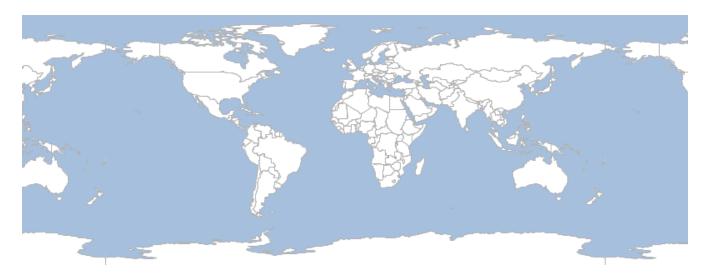




PNG

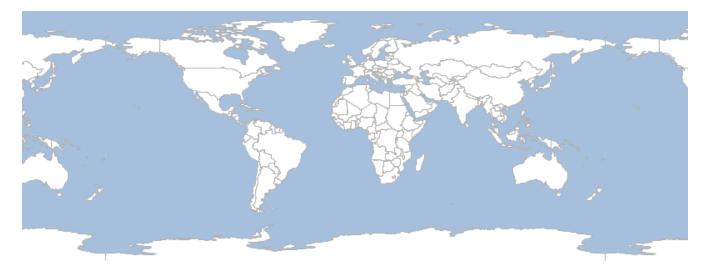
Render a Map to an Image using the PNG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
PNG png = new PNG()
BufferedImage image = png.render(map)
```

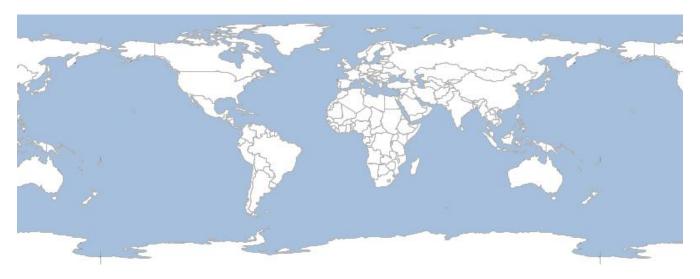


Render a Map to an OutputStream using the PNG Renderer

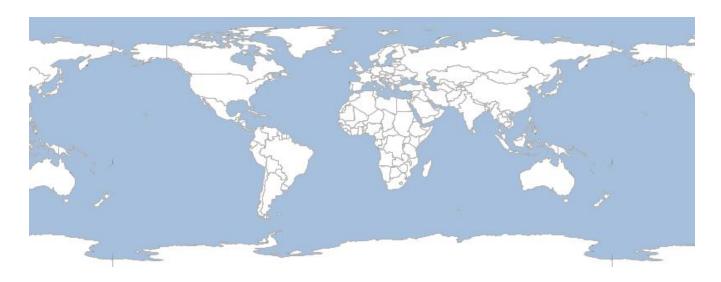
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
PNG png = new PNG()
File file = new File("map.png")
FileOutputStream out = new FileOutputStream(file)
png.render(map, out)
out.close()
```



JPEG

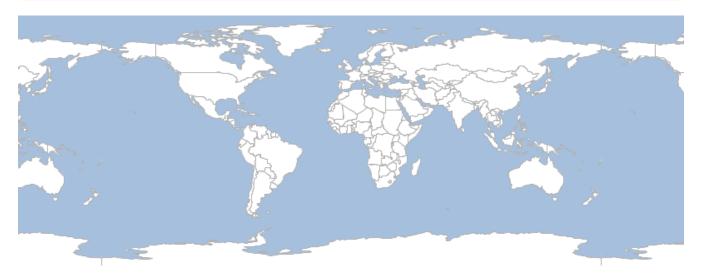


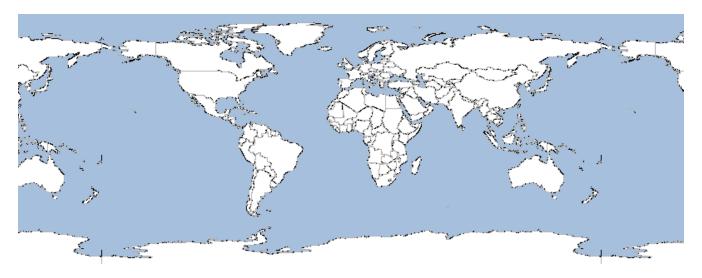
Render a Map to an OutputStream using the JPEG Renderer



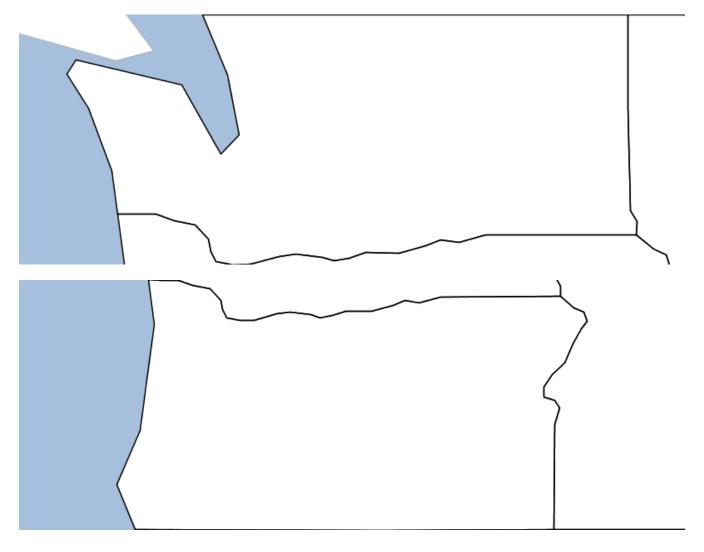
GIF

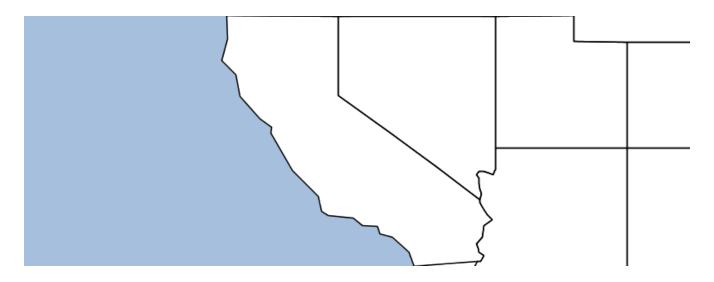
Render a Map to an Image using the GIF Renderer





```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer states = workspace.get("states")
states.style = new Fill("") + new Stroke("black", 1.0)
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
        width: 800,
        height: 300,
        layers: [ocean, countries, states]
)
GIF gif = new GIF()
List images = ["Washington", "Oregon", "California"].collect { String state ->
    map.bounds = states.getFeatures("NAME_1 = '${state}'")[0].bounds
    def image = gif.render(map)
    image
}
File file = new File("states.gif")
gif.renderAnimated(images, file, 500, true)
```

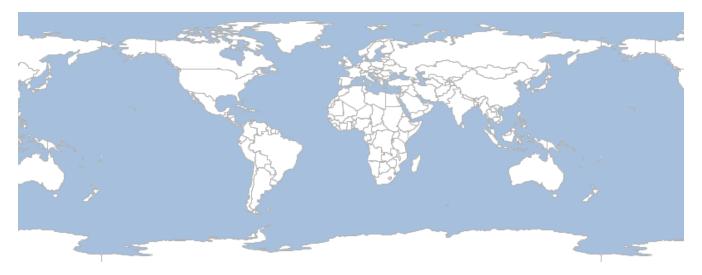




GeoTIFF

Render a Map to an Image using the GeoTIFF Renderer





ASCII

Render a Map to an string using the ASCII Renderer

```
....:^(^..??^^!^:..!(^...(((((^...................^(^..??^
....(:((.....(:((.((((((((......(:
```

Render a Map to an text file using the ASCII Renderer

```
....:^(^..??^^!^:..!(^...(((((^...................^(^..??^
....(:((.....(:((.((((((((......(:
......
```

Base64

Render a Map to an string using the Base64 Renderer

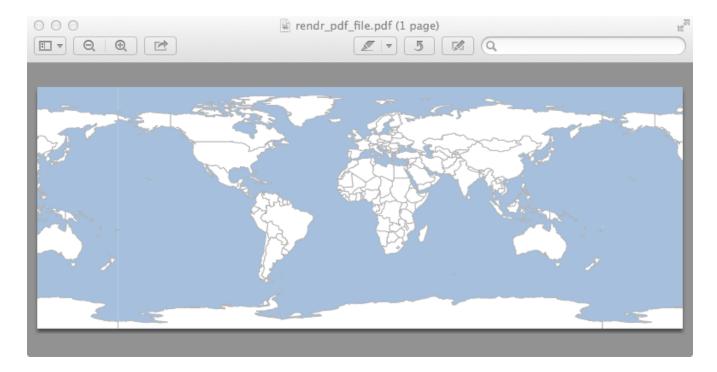
```
image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAyAAAAEsC...
```

Render a Map to an text file using the Base64 Renderer

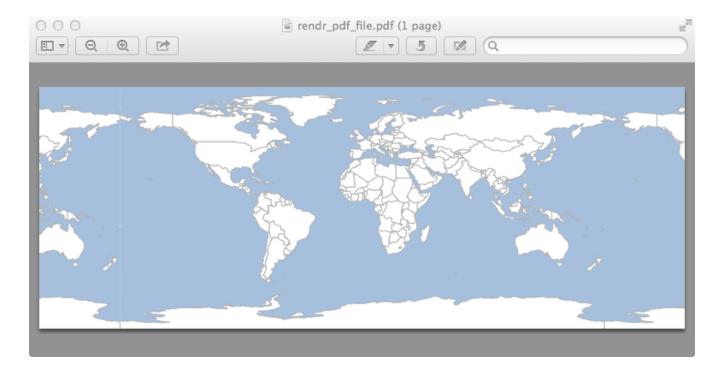
iVBORw0KGgoAAAANSUhEUgAAAyAAAAEsCAYAAAA7Ldc6AACAAE...

PDF

Render a Map to a PDF Document using the PDF Renderer



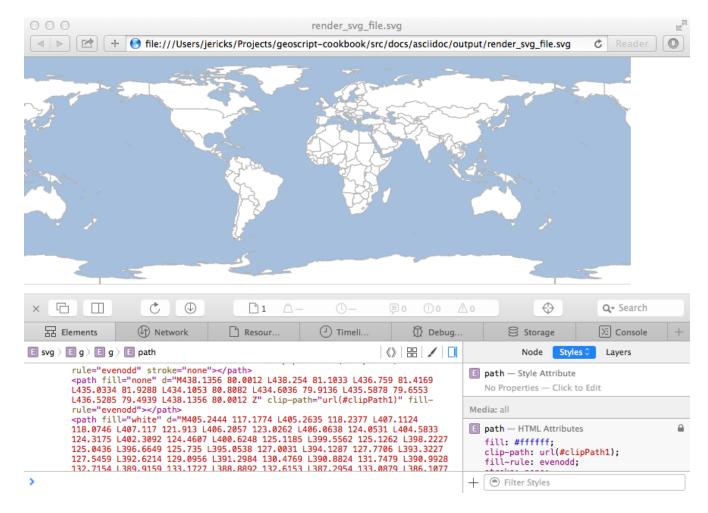
Render a Map to a PDF file using the PDF Renderer



SVG

Render a Map to a SVG Document using the SVG Renderer

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer countries = workspace.get("countries")
countries.style = new Fill("#ffffff") + new Stroke("#b2b2b2", 0.5)
Layer ocean = workspace.get("ocean")
ocean.style = new Fill("#a5bfdd")
Map map = new Map(
    width: 800,
    height: 300,
    layers: [ocean, countries]
)
Svg svg = new Svg()
org.w3c.dom.Document document = svg.render(map)
```



Render a Map to a SVG file using the SVG Renderer



Displaying Maps

Finding Displayers

Get all Displayers

```
List<Displayer> displayers = Displayers.list()
displayers.each { Displayer displayer ->
    println displayer.class.simpleName
}
```

MapWindow Window

Get a Displayer

```
Displayer displayer = Displayers.find("window")
println displayer.class.simpleName
```

Window

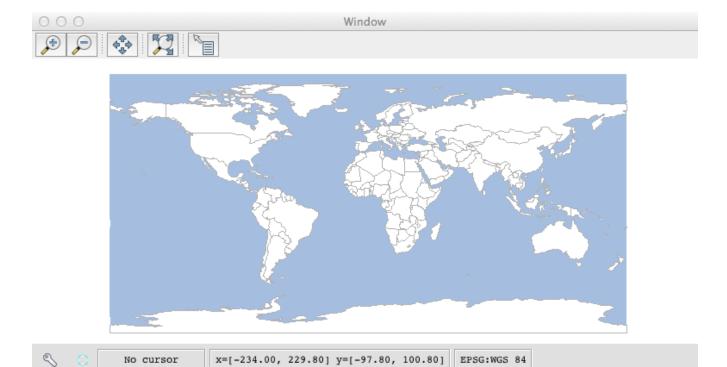
Window

Display a Map in a simple GUI



MapWindow

Display a Map in a interactive GUI



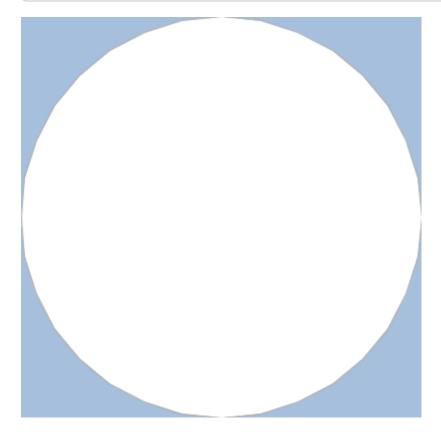
Drawing

The Draw class is an easy way to quickly render a Geometry, a List of Geometries, a Feature, or a Layer to an Image, a File, an OutputStream, or a GUI.

Drawing to a File or GUI

All of the draw methods take a single required parameter but can also take the following optional parameters:

- style = A Style
- bounds = The Bounds
- size = The size of the canvas ([400,350])
- out = The OutputStream, File, or File name. If null (which is the default) a GUI will be opened.
- format = The format ("jpeg", "png", "gif")
- proj = The Projection



Draw a List of Geometries to an OuputStream

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
        point.buffer(distance)
}
File file = new File("geometries.png")
OutputStream outputStream = new FileOutputStream(file)
Draw.draw(geometries, out: outputStream, format: "png")
outputStream.flush()
outputStream.close()
```



Draw a Feature to a file name

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
Feature feature = layer.first(filter: "NAME_1='Washington'")
Draw.draw(feature, bounds: feature.bounds, out: "feature.png")
```



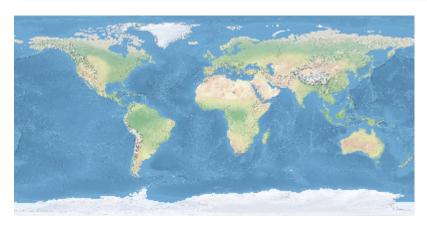
Draw a Layer to a GUI

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
layer.style = new Fill("LightSteelBlue") + new Stroke("LightSlateGrey", 0.25)
Draw.draw(layer, bounds: layer.bounds)
```



Draw a Raster to a File

```
File file = new File("earth.png")
Raster raster = new geoscript.layer.GeoTIFF(new File('src/main/resources/earth.tif'
)).read()
Draw.draw(raster, bounds: raster.bounds, size: [400,200], out: file)
```



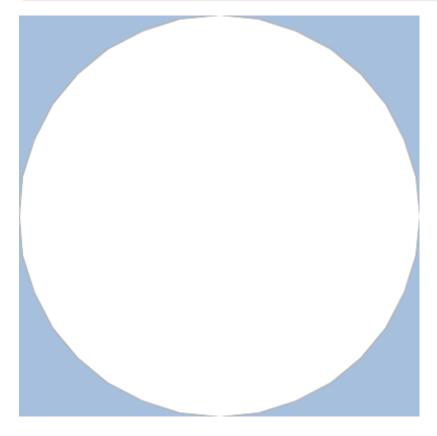
Drawing to an Image

All of the drawToImage methods take a single required parameter but can also take the following optional parameters:

• style = A Style

- bounds = The Bounds
- size = The size of the canvas ([400,350])
- imageType = The format ("jpeg", "png", "gif")
- proj = The Projection

Draw a Geometry to an Image



Draw a List of Geometries to an Image

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
BufferedImage image = Draw.drawToImage(geometries)
```



Draw a Feature to an Image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
Feature feature = layer.first(filter: "NAME_1='Washington'")
BufferedImage image = Draw.drawToImage(feature, bounds: feature.bounds)
```



Draw a Layer to an Image

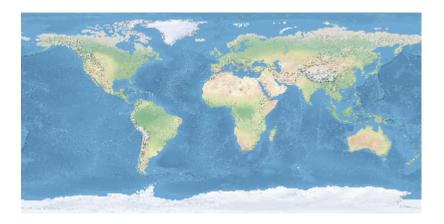
```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
layer.style = new Fill("LightSteelBlue") + new Stroke("LightSlateGrey", 0.25)
BufferedImage image = Draw.drawToImage(layer, bounds: layer.bounds)
```





Draw a Raster to an Image

```
Raster raster = new geoscript.layer.GeoTIFF(new File('src/main/resources/earth.tif'
)).read()
BufferedImage image = Draw.drawToImage(raster, bounds: raster.bounds, size: [400,200])
```

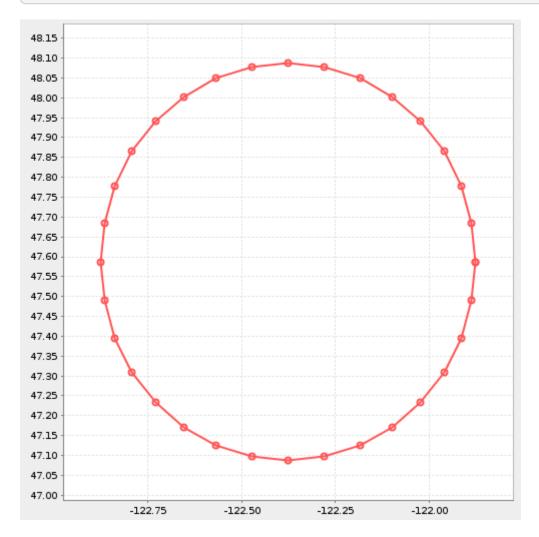


Plotting

Plotting to a File or GUI

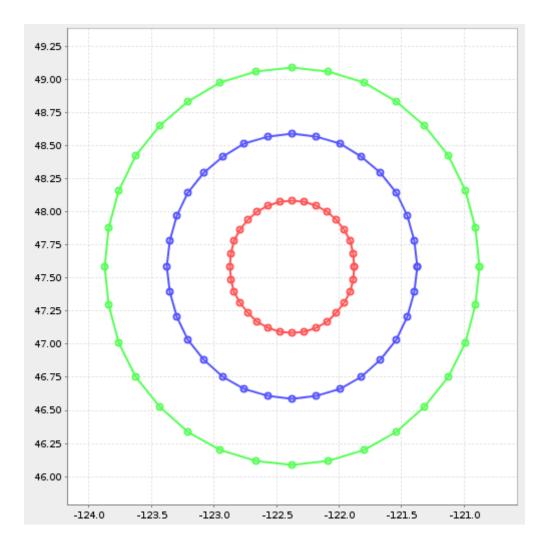
The Plot module can plot a Geometry, a list of Geometries, a Feature, or a Layer to a File, a File name, an OutputStream, or a simple GUI.

```
File file = new File("geometry.png")
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
Plot.plot(geometry, out: file)
```



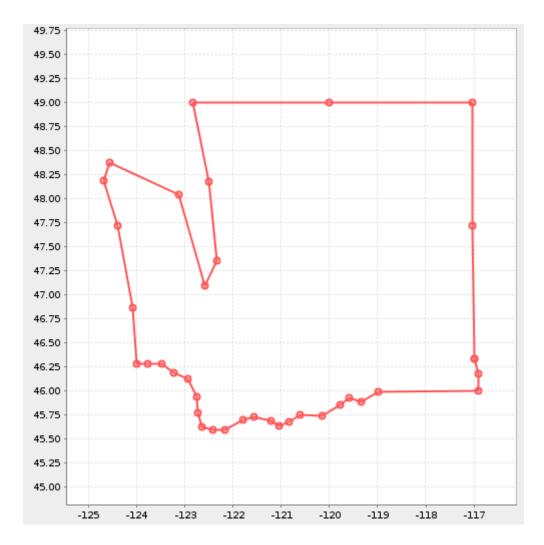
Plot a List of Geometries to an OutputStream

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
        point.buffer(distance)
}
File file = new File("geometries.png")
OutputStream outputStream = new FileOutputStream(file)
Plot.plot(geometries, out: outputStream)
outputStream.flush()
outputStream.close()
```



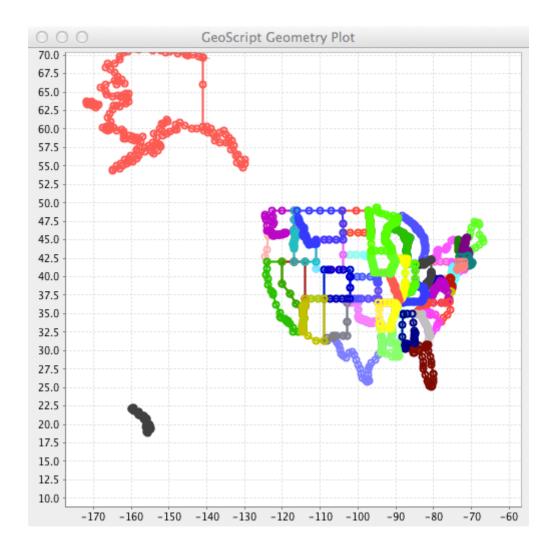
Plot a Feature to a File name

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
Feature feature = layer.first(filter: "NAME_1='Washington'")
Plot.plot(feature, out: "feature.png")
```



Plot a Layer to a GUI

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
Plot.plot(layer)
```

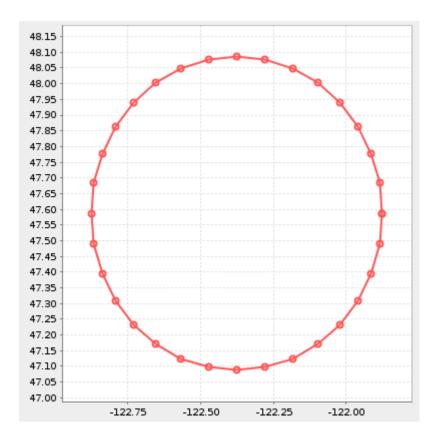


Plotting to an Image

The Plot module can plot a Geometry, a list of Geometries, a Feature, or a Layer to an image.

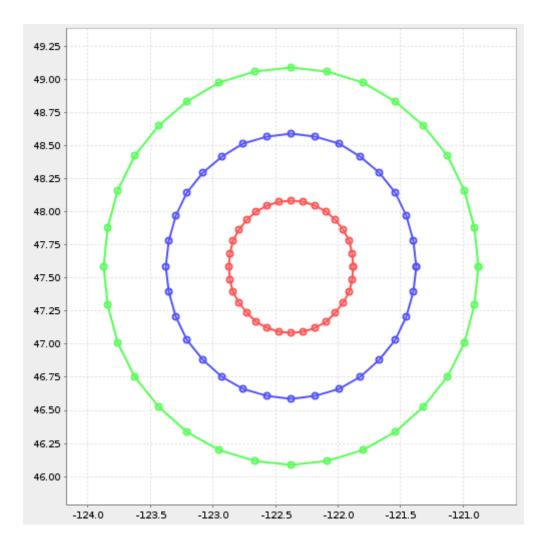
Plot a Geometry to an Image

```
Geometry geometry = new Point(-122.376, 47.587).buffer(0.5)
BufferedImage image = Plot.plotToImage(geometry, size: [400,400],)
```



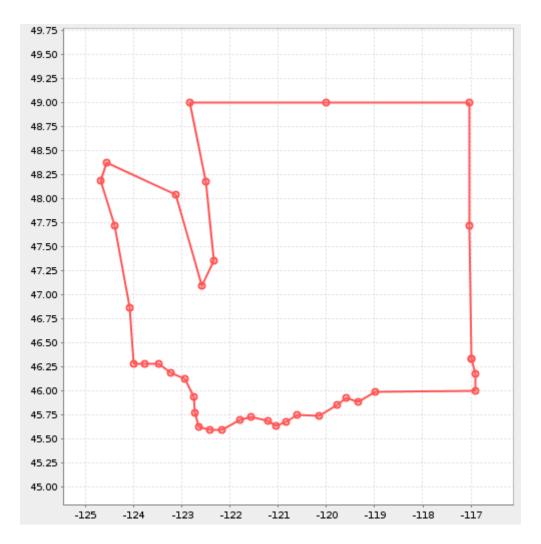
Plot a List of Geometries to an Image

```
Point point = new Point(-122.376, 47.587)
List geometries = [1.5, 1.0, 0.5].collect { double distance ->
    point.buffer(distance)
}
BufferedImage image = Plot.plotToImage(geometries)
```



Plot a Feature to an Image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
Feature feature = layer.first(filter: "NAME_1='Washington'")
BufferedImage image = Plot.plotToImage(feature, bounds: feature.bounds)
```



Plot a Layer to an Image

```
Workspace workspace = new GeoPackage('src/main/resources/data.gpkg')
Layer layer = workspace.get("states")
BufferedImage image = Plot.plotToImage(layer, bounds: layer.bounds)
```

