

gMapIoKml

Generating KML scripts for displaying
placemarks, polylines and polygons in Google
Earth

Rev. 1.1

Contents

Introduction	4
1. Create a kml-file	6
2. Close a kml file	6
3. Covert color and transparency values to the kml format.....	6
4. Set graphical element Styles	7
5. Draw a Placemark (point)	9
6. Draw a Polyline	10
7. Draw a Polygon	10
8. Example of using the gMapIoKml library (gMapIoKmlTest script)	12
Citation.....	15

Revision History

Revision	Date	Description of Changes
1.0	2021-06-12	First issue document// Ivan D.
1.1	2024-03-25	Edited the text, prepared English and Russian versions // Ivan D., Margarita B.

Tables list

Table 0.1 gMapIoKml functions 4

Figures list

Figure 0.1 The gMapIoKml library structure 5

Figure 8.1 gMapIoKmlTest script result (the 112.kml file is opened in Google Earth).....12

Introduction

The gMapIoKml library contains MatLab/GNU Octave ([Table 0.1](#),) functions for generating kml-scripts [1, 2, 3], which are used to display graphical objects in Google Earth, Google Maps or any GIS. The library functions write KML text "commands" to a *.kml file format.

When creating the library, the goal was not to use all the features and syntax of KML, but to perform the necessary minimum of actions to generate a kml file while minimizing the time spent on studying the gMapIoKml library functions. The library functions allow you to perform the following sequence of actions with a kml file:

- 1) create the kml-file and write necessary "file start tags" to the file (gMapIoKmlOpen);
- 2) set the display styles of graphical elements – placemark, line, polygon, description balloon, etc. in kml-file (gMapIoKmlSetStyle);
- 3) write the "commands" in the kml file: draw a placemark, draw a polyline, draw a polygon (gMapIoKmlSetPoint, gMapIoKmlSetPolyline, gMapIoKmlSetPolygon);
- 4) write necessary "end-of-file tags" to the file and close the kml file (gMapIoKmlClose).

The gMapIoKmlColor function is used to convert color and transparency values to the format used in kml. The converted color and transparency values are used to set styles (gMapIoKmlSetStyle command).

Table 0.1 gMapIoKml functions

Function name	Function description
gMapIoKmlOpen	Create a kml-file
gMapIoKmlColor	Convert color and transparency values to the kml format
gMapIoKmlSetStyle	Set graphical element Styles
gMapIoKmlSetPoint	Draw a Placemark (point)
gMapIoKmlSetPolyline	Draw a Polyline
gMapIoKmlSetPolygon	Draw a Polygon
gMapIoKmlClose	Close a kml-file
gMapIoKmlTest	gMapIoKml script with an example of using the library functions

When creating the library, the code of the kml_line function was used [4].

Currently, reading and writing kml-files functions are included in the standard Mapping Toolbox library.

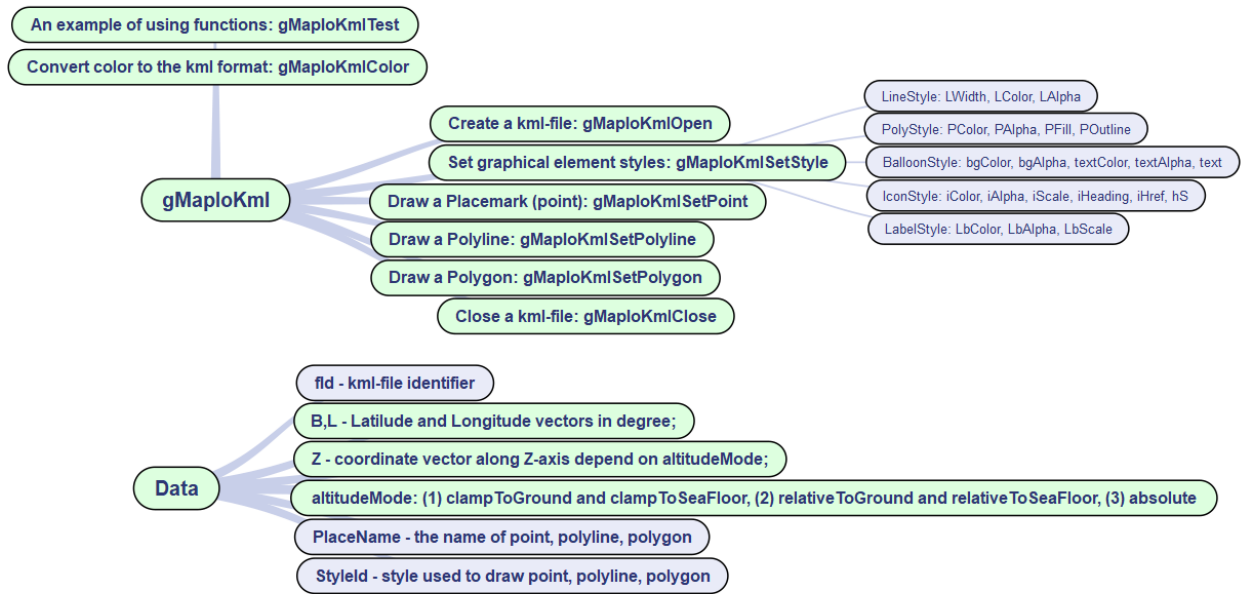


Figure 0.1 The gMapIoKml library structure

1. Create a kml-file

function fId=gMapIoKmlOpen(FileName)

Creates a kml-file named FileName and writes "kml file start tags" to it.

Parameters:

fId – script file identifier;

FileName – the name of kml-file.

Function Example:

```
>> fId=gMapIoKmlOpen('c:\temp\112.kml');
```

2. Close a kml file

function status=gMapIoKmlClose(fId)

Writes "end of kml file tags" to the kml file and closes the file with the fId identifier.

Parameters:

fId – script file identifier;

status – a variable that takes the value '0' if the file close operation was successful, the value '-1' otherwise.

Function Example:

```
>> gMapIoKmlClose(fId);
```

3. Covert color and transparency values to the kml format

function [LColor,LAlpha]=gMapIoKmlColor(LColor,LAlpha)

Converts Color and transparency values to the format used in the kml-file from other formats. The converted color and transparency values are used when setting styles (gMapIoKmlSetStyle function).

Parameters:

LColor – color that can be set in three ways: (1) a sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff'; (2) the letters 'r', 'g', 'b'; (3) a vector of three numeric values from 0 to 255 encoding red-green-blue colors, for example [255 255 0];

LAlpha – transparency that can be set in two ways: (1) a sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity; (2) a numeric value from 0 to 255.

The output values of LColor and LAlpha are returned as a sequence of characters (for example, '00ffff' or '9f') that is the format used in the kml-file.

Function Example:

```
>> [LColor,LAlpha]=gMapIoKmlColor('r',255);
```

4. Set graphical element Styles

function gMapIoKmlSetStyle(fId,StyleId,varargin)

Writes Style tags for graphical elements to a kml-file. In fact, the function creates a style named StyleId, which contains a display style for five elements: placemarks (IconStyle), description balloons (BalloonStyle), polylines (LineStyle), polygons (PolyStyle) and names of objects when displayed in 3D (LabelStyle). The values colorMode='normal', gx:labelVisibility=0 are implicitly set in the function code.

Parameters:

fId – script file identifier;

StyleId – style unique identificator (string or number); examples: 'Stl09', 28, 'RedStyle';

varargin – cell array that can be used to set styles for the following elements: 'LineStyle', 'BalloonStyle', 'IconStyle', 'PolyStyle', 'LabelStyle'; for example: {'LineStyle',4,'r','ff'}. If the value of any parameter (the parameters for style elements are listed below) is set as an empty array [], then the kml code corresponding to this parameter will not be written to the kml file. The color code (Color) and fill transparency (Alpha) must be set as empty arrays at the same time so that the corresponding kml code is not written to the kml-file.

'LineStyle' specifies the drawing style (color, color mode, and line width) for all line geometry. Line geometry includes the outlines of outlined polygons and the extruded "tether" of Placemark icons (if extrusion is enabled). 'LineStyle' element parameters:

LWidth – line width in pixels; examples: '05', 4;

LColor – line color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

LAlpha – line color transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

'PolyStyle' specifies the drawing style for all polygons, including polygon extrusions (which look like the walls of buildings) and line extrusions (which look like solid fences) [1]. 'PolyStyle' element parameters:

PColor – polygon fill color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

PAlpha – polygon transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

PFill – boolean value that specifies whether to fill the polygon or not; example: '1',0;

POutline – boolean value that specifies whether to outline the polygon or not. Polygon outlines use the current LineStyle; example: '1',0;

'BalloonStyle' specifies how the description balloon for placemarks is drawn. The <bgColor>, if specified, is used as the background color of the balloon. 'BalloonStyle' element parameters:

bgColor – balloon background color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

bgAlpha – balloon background color transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

textColor – balloon text color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

textAlpha – balloon text transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

text – text to be displayed in the balloon. Text example:

```
<![CDATA[<b><font color="#CC0000" size="+3">${name}</b> <br> ${description}]]>
```

A detailed description of the 'BalloonStyle' element is given in [5].

'IconStyle' specifies how icons for point Placemarks are drawn, both in the Places panel and in the 3D viewer of Google Earth. The <Icon> element specifies the icon image. The <scale> element specifies the x, y scaling of the icon. The color specified in the <color> element of <IconStyle> is blended with the color of the <Icon>. 'IconStyle' element parameters:

iColor – icon color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

iAlpha – icon transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

iScale – icon scale parameter (default=1);

iHeading – icon rotation that is set in the compass direction in degrees; default=0 (North); values range from 0 to ± 180 degrees;

iHref – HTTP address or a local file specification used to load an icon; example:
`http://maps.google.com/mapfiles/kml/pal3/icon21.png`

hS – specifies the position within the Icon that is "anchored" to the <Point> specified in the Placemark
hS={hS_x,hS_y,hS_xunits,hS_yunits} vector containing the position (the coordinates) within the Icon that is "anchored" to the <Point> specified in the Placemark. The x and y values can be set using three different units: (1) pixels with the origin of the coordinate system in the bottom left corner of the icon ("pixels"); (2) fractions of the icon with the origin of the coordinate system in the bottom left corner of the icon ("fraction"); (3) pixels with the origin of the coordinate system in the top right corner of the icon ("insetPixels"). The x and y positions can be specified in different ways – for example, x can be in pixels and y can be a fraction.

hS_x – offset along the x-axis;

hS_y – offset along the y-axis;

xunits – units for hS_x ("fraction", "pixels" or "insetPixels");

yunits – units for hS_y ("fraction", "pixels" or "insetPixels");

'**LabelStyle**' specifies how the <name> of a Feature is drawn in the 3D viewer. A custom color, color mode, and scale for the label (name) can be specified. 'LabelStyle' element parameters:

LbColor – label color that is set by the sequence of characters bbgrr, where bb is blue, gg is green, rr is red, for example '00ffff';

LbAlpha – label color transparency that is set by the sequence of characters from '00' in case of full transparency to the value of 'ff' in case of opacity;

LbScale – label font size.

Function Example:

```
>> gMapIoKmlSetStyle(fId,'st001',{ 'LineStyle',4,'r','ff'},{ 'BalloonStyle','c','0f','b','ff','Area01'});
```

```
>> gMapIoKmlSetStyle(fId,'style02',{ 'LineStyle',4,'r','ff'},{ 'PolyStyle','b','5f','1','0'});
```

5. Draw a Placemark (point)

function gMapIoKmlSetPoint(fId,B,L,Z,altitudeMode,PlaceName,StyleId,Descript)

Writes tags and data to a kml-file – to draw a Placemark (point) on the map. The value extrude=0 is implicitly set in the function code.

Parameters:

fId – script file identifier;

B – latitude in degree (one numeric value);

L – longitude in degree (one numeric value);

Z – coordinate along Z-axis the display of which depends on how altitude components in the <coordinates> element are interpreted (altitudeMode); if the Z value is set as an empty array [], then the altitude value is 0 (clampToGround mode is set);

altitudeMode – specifies how altitude components in the <coordinates> element are interpreted. Possible values are:

- (1) clampToGround or clampToSeaFloor – altitude value is 0,
- (2) relativeToGround or relativeToSeaFloor – relative to the actual ground elevation of a particular location or above the sea floor,
- (3) absolute – relative to sea level, regardless of the actual elevation of the terrain beneath the element (an absolute altitude value relative to the ellipsoid).

PlaceName – name of the placemark; examples: 'Point002', 28;

StyleId – placemark style unique identifier (IconStyle, BalloonStyle, LabelStyle) set as a string or a numeric value when setting styles using the gMapIoKmlSetStyle function; examples: 'Stl09', 28, 'RedStyle';

Descript – placemark description text to be displayed in the balloon; examples: 'seismic survey, area B005',
'<![CDATA[Multichannel Seismic data

]]>'

Function Example:

```
>> gMapIoKmlSetPoint(fId,65.1,141,10,2,'Line023','B1','1234567890');
```

6. Draw a Polyline

function gMapIoKmlSetPolyline(fId,B,L,Z,altitudeMode,PlaceName,LineStyleId,Descript)

Writes tags and data to a kml-file – to draw a Polyline on the map. The values `extrude=0` and `tessellate=1` are implicitly set in the function code.

Parameters:

fId – script file identifier;

B – latitude vector in degree (one numeric value);

L – longitude vector in degree (one numeric value);

Z – coordinate along Z-axis the display of which depends on how altitude components in the `<coordinates>` element are interpreted (altitudeMode); if the Z value is set as an empty array `[ ]`, then the altitude value is 0 (clampToGround mode is set);

altitudeMode – specifies how altitude components in the `<coordinates>` element are interpreted. Possible values are:

- (1) `clampToGround` or `clampToSeaFloor` – altitude value is 0,
- (2) `relativeToGround` or `relativeToSeaFloor` – relative to the actual ground elevation of a particular location or above the sea floor,
- (3) `absolute` – relative to sea level, regardless of the actual elevation of the terrain beneath the element (an absolute altitude value relative to the ellipsoid).

PlaceName – name of the polyline; examples: 'Line002', 28;

LineStyleId – the line style unique identifier (LineStyle, BalloonStyle, LabelStyle) set as a string or a numeric value when setting styles using the `gMapIoKmlSetStyle` function; examples: 'Stl09', 28, 'RedStyle';

Descript – polyline description text to be displayed in the balloon; examples: 'seismic survey, area B005',
'<![CDATA[Multichannel Seismic data

]]>'

Function Example:

```
>> gMapIoKmlSetPolyline(fId,[65.1 65.2 65.3],[141 141 142],[],1,'Line023','st001','');
```

7. Draw a Polygon

function

gMapIoKmlSetPolygon(fId,B,L,Z,altitudeMode,PlaceName,LineStyleId,Descript,varargin)

Writes tags and data to a kml-file – to draw a Polygon on the map. The values `extrude=0` and `tessellate=1` are implicitly set in the function code. A polygon is defined by an outer boundary and 0 or more inner boundaries. The `<coordinates>` for polygons must be specified in counterclockwise order. The last point must be equal the first point. The inner boundaries are defined by LinearRings.

When a polygon is extruded, its boundaries are connected to the ground to form additional polygons, which gives the appearance of a building or a box. Extruded polygons use <PolyStyle> for their color, color mode, and fill.

Polygons follow the "right-hand rule," which states that if you place the fingers of your right hand in the direction in which the coordinates are specified, your thumb points in the general direction of the geometric normal for the polygon. (In 3D graphics, the geometric normal is used for lighting and points away from the front face of the polygon.) Since Google Earth fills only the front face of polygons, you will achieve the desired effect only when the coordinates are specified in the proper order. Otherwise, the polygon will be gray.

Parameters:

fId – script file identifier;

B – latitude in degree (one numeric value);

L – longitude in degree (one numeric value);

Z – coordinate along Z-axis the display of which depends on how altitude components in the <coordinates> element are interpreted (altitudeMode); if the Z value is set as an empty array [], then the altitude value is 0 (clampToGround mode is set);

altitudeMode – specifies how altitude components in the <coordinates> element are interpreted. Possible values are:

- (1) clampToGround or clampToSeaFloor – altitude value is 0,
- (2) relativeToGround or relativeToSeaFloor – relative to the actual ground elevation of a particular location or above the sea floor,
- (3) absolute – relative to sea level, regardless of the actual elevation of the terrain beneath the element (an absolute altitude value relative to the ellipsoid).

PlaceName – name of the polygon; examples: 'Polygon002', 28;

LineStyleId – the polygon style unique identifier (LineStyle, BalloonStyle, LabelStyle) set as a string or a numeric value when setting styles using the gMapIoKmlSetStyle function; examples: 'Stl09', 28, 'RedStyle';

Descript – polygon description text to be displayed in the balloon; examples: 'seismic survey, area B005', '<![CDATA[Multichannel Seismic data

]]>'

varargin – array of cells that defines the polygon “inner boundaries” (holes inside the polygon shown in [Figure 8.1](#)); each cell contains a matrix of three row vectors (latitude, longitude and altitude Z -- {B,L,Z}) or two row vectors (latitude and longitude -- {B,L}).

Function Example:

```
>> gMapIoKmlSetPolygon(fId,[64 64.5 63.5 64],[143 140 139 143],[,],1,'Polygon01','style02','Polygon
example with style02',{[64 64 63.8],[140.8 140.25 140.3],[,]},{[63.98 63.62 63.78],[140 139.30
140.19],[,]});
```

8. Example of using the gMapIoKml library (gMapIoKmlTest script)

The example of using the gMapIoKml library functions is written in the gMapIoKmlTest script, which is the part of the gMapIoKml library. The script text and the code written to the kml-file by this script are shown below. **Figure 8.1** shows the kml-file content in Google Earth.

```
>> fId=gMapIoKmlOpen('c:\temp\112.kml');
>>
>>gMapIoKmlSetStyle(fId,'style01',{ 'LineStyle',4,'r','ff'},{ 'BalloonStyle','c','0f','b','0f','<![CDATA[<b><font color="#CC0000" size="-3">${name}</b> <br> ${description}]]>'});
>> gMapIoKmlSetStyle(fId,'style02',{ 'LineStyle',4,'r','ff'},{ 'PolyStyle','b','5f','1','0'});
>> gMapIoKmlSetPoint(fId,64.85,140.34,10,2,'Line023','style01','Point example with style01');
>> gMapIoKmlSetPolyline(fId,[65.1 65.2 65.5],[141 140.5 142],[],1,'Line023','style01','Line example with style01');
>> gMapIoKmlSetPolygon(fId,[64 64.5 63.5 64],[143 140 139 143],[],1,'Polygon01','style02','Polygon example with style02',{[64 64 63.8],[140.8 140.25 140.3],[],[[63.98 63.62 63.78],[140 139.30 140.19],[[]]});
gMapIoKmlClose(fId);
```

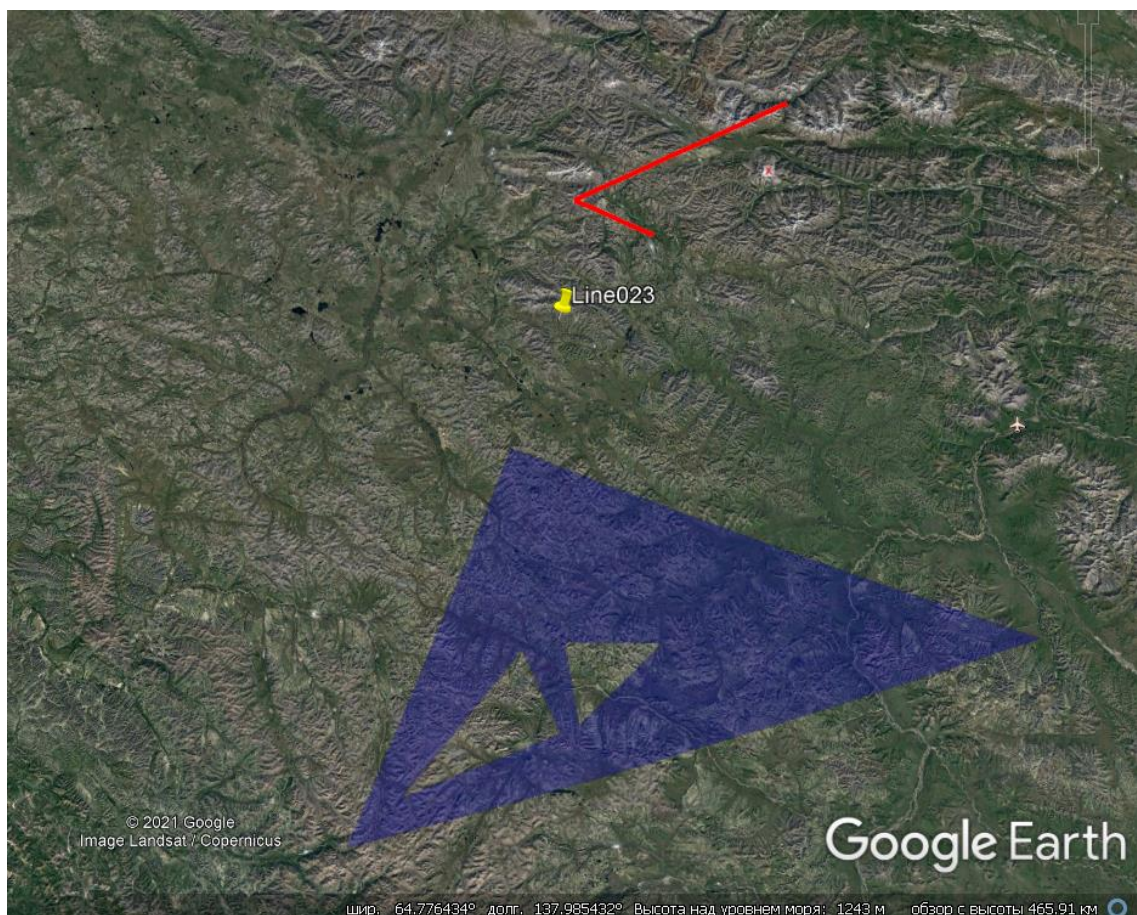


Figure 8.1 gMapIoKmlTest script result (the 112.kml file is opened in Google Earth)

The code of 112.kml file was created as a result of executing the gMapIoKmlTest script. The 112.kml-file contents are shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
  <name>112.kml</name>
  <Style id="style01">
    <LineStyle>
      <colorMode>normal</colorMode>
      <color>#ff0000ff</color>
      <width>4</width>
      <gx:labelVisibility>0</gx:labelVisibility>
    </LineStyle>
    <BalloonStyle>
      <bgColor>#0fffff00</bgColor>
      <textColor>#0ff00000</textColor>
      <text><![CDATA[<b><font color="#CC0000" size="-3">${name}</b> <br> ${description}]]></text>
      <displayMode>default</displayMode>
    </BalloonStyle>
  </Style>
  <Style id="style02">
    <LineStyle>
      <colorMode>normal</colorMode>
      <color>#ff0000ff</color>
      <width>4</width>
      <gx:labelVisibility>0</gx:labelVisibility>
    </LineStyle>
    <PolyStyle>
      <colorMode>normal</colorMode>
      <color>#5fff0000</color>
      <fill>1</fill>
      <outline>0</outline>
    </PolyStyle>
  </Style>
  <Placemark>
    <name>Line023</name>
    <description>Point example with style01</description>
    <styleUrl>#style01</styleUrl>
    <Point>
      <extrude>0</extrude>
      <altitudeMode>relativeToGround</altitudeMode>
      <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
      <coordinates>
        140.34,64.85,10
      </coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name>Line023</name>
    <description>Line example with style01</description>
    <styleUrl>#style01</styleUrl>
    <LineString>
      <extrude>0</extrude>
      <tessellate>1</tessellate>
      <altitudeMode>clampToGround</altitudeMode>
      <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
      <coordinates>
        141,65.1,0,140.5,65.2,0,142,65.5,0
      </coordinates>
    </LineString>
  </Placemark>
  <Placemark>
    <name>Polygon01</name>
    <description>Polygon example with style02</description>
    <styleUrl>#style02</styleUrl>
    <Polygon>
      <extrude>0</extrude>
      <tessellate>1</tessellate>
      <altitudeMode>clampToGround</altitudeMode>
      <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
      <outerBoundaryIs>
        <LinearRing>
          <altitudeMode>clampToGround</altitudeMode>
          <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
```

```

        <coordinates>
            143,64,0,140,64.5,0,139,63.5,0,143,64,0
        </coordinates>
    </LinearRing>
</outerBoundaryIs>
<innerBoundaryIs>
    <LinearRing>
        <altitudeMode>clampToGround</altitudeMode>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>
            140.8,64,0,140.25,64,0,140.3,63.8,0
        </coordinates>
    </LinearRing>
</innerBoundaryIs>
<innerBoundaryIs>
    <LinearRing>
        <altitudeMode>clampToGround</altitudeMode>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>
            140,63.98,0,139.3,63.62,0,140.19,63.78,0
        </coordinates>
    </LinearRing>
</innerBoundaryIs>
</Polygon>
</Placemark>
</Document>
</kml>

```

Citation

1. <https://developers.google.com/kml/documentation/kmlreference>
2. https://ge0mlib.com/papers/File_Formats/KML_v1.0_cookbook.pdf
3. https://ge0mlib.com/papers/File_Formats/KML_v2.1_Reference_An_OGC_Best_Practice.pdf
4. <https://www.mathworks.com/matlabcentral/fileexchange/34006-kml-line-plot>
5. <https://developers.google.com/kml/documentation/kmlreference#balloonstyle>