KML

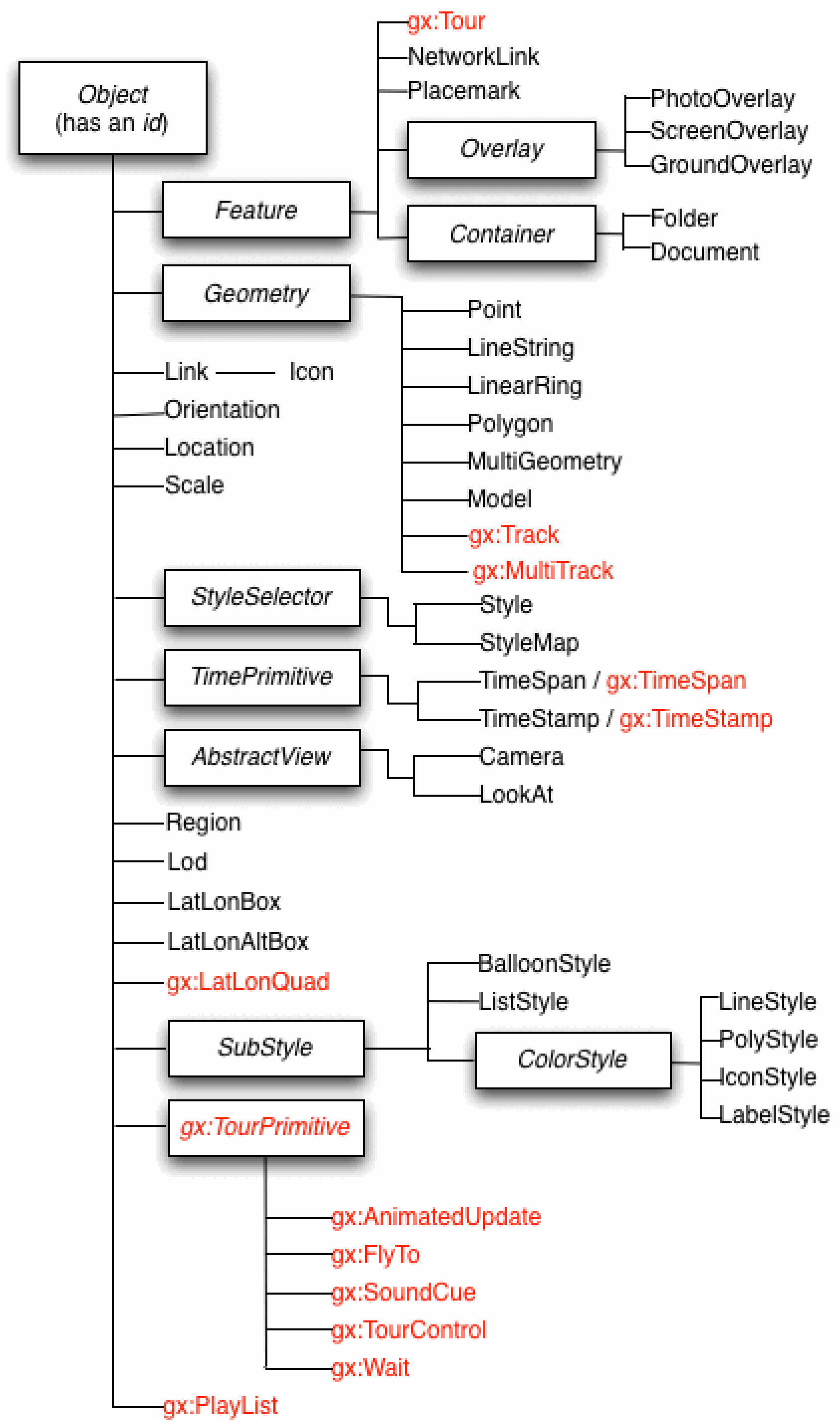
KML Reference

This section contains an alphabetical reference for all KML elements defined in KML Version 2.2, as well as elements in the Google extension namespace. The class tree for KML elements is shown below. In this diagram, elements to the right on a particular branch in the tree are *extensions* of the elements to their left. For example, Placemark is a special kind of *Feature*. It contains all of the elements that belong to *Feature*, and it adds some elements that are specific to the Placemark element.

KML is an open standard officially named the OpenGIS® KML Encoding Standard (OGC KML). It is maintained by the Open Geospatial Consortium, Inc. (OGC). The complete specification for OGC KML can be found at http://www.opengeospatial.org/standards/kml/.

The complete XML schema for KML is located at http://schemas.opengis.net/kml/.

Note: Click an element name in this diagram to jump to its entry in the reference section.



Note that abstract elements (shown in boxes in the diagram) are not actually used in KML files. They are a useful way for a single element to serve as the programmatic foundation for multiple similar (but different) derived elements. Understanding this object-oriented hierarchy is also a good way for you to learn KML, since you can easily see groupings of related elements.

All elements derived from *Object* can have an **id** assigned to them. This **id** is used by the KML update mechanism (see <Update>) for files loaded with a NetworkLink. It is also used by shared styles (see <Style>). The **id** is a standard XML ID.

Because KML is an XML grammar and file format, tag names are case-sensitive and must appear exactly as shown here. If you're familiar with XML, you will also be interested in the KML 2.2 Schema. When you are editing KML text files, you can load this Schema into any XML editor and validate your KML code with it.

Tip: Viewing KML for Google Earth Features

Here is a handy feature of Google Earth that makes it easy to view the KML file for any Feature. In Google Earth, you can right-click a Feature in the Places panel and copy it. To view the corresponding KML for the copied object, open your favorite text editor and paste the selection into it.

Compatibility

KML versions have a double numbering system: `majorVersion.minorVersion`. All versions with the same `majorVersion` are compatible. For this reason, if you change the namespace to "2.2" (that is, `xmlns="http://www.opengis.net/kml/2.2"`), all KML 2.1 files validate in the KML 2.2 schema.

About this reference

Each reference entry includes a Syntax section that lists the elements contained in the main element. This Syntax section is an informal listing and uses simple shorthand to summarize the elements. This section also contains the following:

- *default values* for each element (or ellipses if it is a complex element or if there is no default value)
- the *type* of the value (see KML Fields)

The Syntax section can be copied and used as a template for any non-abstract element in a KML file.

KML Extension Namespace and the gx prefix

The OGC KML standard provides a mechanism for extensions - additional elements that contain information beyond what is available in the standard (learn more about XML namespaces at w3.org). With the launch of Google Earth 5.0, Google has provided extensions to KML to support a number of new features. These extensions use the **gx** prefix and the following namespace URI:

```
xmlns:gx="http://www.google.com/kml/ext/2.2"
```

This namespace URI must be added to the `<kml>` element in any KML file using **gx**-prefixed elements:

```
<kml xmlns="http://www.opengis.net/kml/2.2"
 xmlns:gx="http://www.google.com/kml/ext/2.2">
```

Extensions to KML may not be supported in all geo-browsers. If your browser doesn't support particular extensions, the data in those extensions should be silently ignored, and the rest of the KML file should load without errors.

Elements that currently use the **gx** prefix are:

- gx:altitudeMode
- gx:altitudeOffset
- gx:angles
- gx:AnimatedUpdate
- gx:balloonVisibility
- gx:coord
- gx:delayedStart
- gx:drawOrder
- gx:duration
- gx:FlyTo
- gx:flyToMode
- gx:h
- gx:interpolate
- gx:LatLonQuad
- gx:MultiTrack
- gx:outerColor
- gx:outerWidth
- gx:Playlist
- gx:playMode
- gx:SoundCue
- gx:TimeSpan
- gx:TimeStamp
- gx:Tour
- gx:TourControl
- gx:TourPrimitive
- gx:Track

- gx:ViewerOptions
- gx:option
- gx:w
- gx:Wait
- gx:x
- gx:y

The complete XML schema for elements in this extension namespace is located at http://code.google.com/apis/kml/schema/kml22gx.xsd.

KML fields

KML uses common XML types such as *boolean, string, double, float,* and *int*. In addition, it defines a number of field element types. The following table lists some of the most commonly used types defined in KML and links to sample elements that use them:

Field Type	***Value***	***Example Use***
altitudeModeEnum	clampToGround, relativeToGround, absolute	See <LookAt> and <Region>
angle90	a value ≥−90 and ≤90	See <latitude> in <Model>
anglepos90	a value ≥0 and ≤90	See <tilt> in <LookAt>
angle180	a value ≥−180 and ≤180	See <longitude> in <Model>
angle360	a value ≥−360 and ≤360	See <heading>, <tilt>, and <roll> in <Orientation>
color	hexBinary value: *aabbggrr*	See any element that extends *<ColorStyle>*
colorModeEnum	normal, random	See any element that extends *<ColorStyle>*
dateTime	*dateTime, date, gYearMonth, gYear*	See <TimeSpan> and <TimeStamp>
displayModeEnum	default, hide	See <BalloonStyle>
gridOrigin	lowerLeft, upperLeft	See <PhotoOverlay>
refreshModeEnum	onChange, onInterval, onExpire	See <Link>
shapeEnum	rectangle, cylinder, sphere	See <PhotoOverlay>
styleStateEnum	normal, highlight	See <StyleMap>
unitsEnum	fraction, pixels, insetPixels	See <hotSpot> in <IconStyle>, <ScreenOverlay>

vec2	x=*double* xunits=*kml:unitsEnum*
	y=*double* yunits=*kml:unitsEnum*
		See <hotSpot> in <IconStyle>, <ScreenOverlay>

viewRefreshEnum	never, onRequest, onStop, onRegion	See <Link>

<AbstractView>

Syntax

```
<!-- abstract element; do not create -->
<!-- AbstractView -->                    <!-- Camera, LookAt -->
  <!-- extends Object -->
  <TimePrimitive>...</TimePrimitive>     <!-- gx:TimeSpan or gx:TimeStamp -->
  <gx:ViewerOptions>
      <gx:option name=" " enabled=boolean />   <!--
name="streetview", "historicalimagery",
                                                         or "sunlight" -->
  </gx:ViewerOptions>
<-- /AbstractView -->
```

Description

This is an abstract element and cannot be used directly in a KML file. This element is extended by the <Camera> and <LookAt> elements.

Extends

- <Object>

Elements Specific to AbstractView

<gx:ViewerOptions>

This element enables special viewing modes in Google Earth 6.0 and later. It has one or more `<gx:option>` child elements. The `<gx:option>` element has a `name` attribute and an `enabled` attribute. The `name` specifies one of the following: Street View imagery ("streetview"), historical imagery ("historicalimagery"), and sunlight effects for a given time of day ("sunlight"). The `enabled` attribute is used to turn a given viewing mode on or off.

Extended By

- <Camera>
- <LookAt>

<gx:altitudeMode>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:altitudeMode>clampToGround</gx:altitudeMode>
  <!-- gx:altitudeModeEnum: relativeToSeaFloor, clampToSeaFloor,
relativeToGround, clampToGround, absolute -->
```

Description

Can be used instead of the OGC KML standard `<altitudeMode>` element, and accepts the following values in addition to the standard altitudeMode values:

- **relativeToSeaFloor** - Interprets the altitude as a value in meters above the sea floor. If the KML feature is above land rather than sea, the altitude will be interpreted as being above the ground.
- **clampToSeaFloor** - The altitude specification is ignored, and the KML feature will be positioned on the sea floor. If the KML feature is on land rather than at sea, **clampToSeaFloor** will instead clamp to ground.

As with `<altitudeMode>`, `<gx:altitudeMode>` affects:

- the altitude coordinate within the `<coordinates>` element
- `<minAltitude>` and `<maxAltitude>` within `<LatLonAltBox>`
- `<altitude>` within `<Location>`, `<GroundOverlay>`, and `AbstractView` (`<LookAt>` and `<Camera>`).

More information about altitude modes is available in the Altitude Modes chapter of the **KML Developer's Guide**.

Example

altitudemode_reference.kml

```
<?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">   <!-- required when using gx-
prefixed elements -->

<Placemark>
  <name>gx:altitudeMode Example</name>
  <LookAt>
    <longitude>146.806</longitude>
    <latitude>12.219</latitude>
    <heading>-60</heading>
    <tilt>70</tilt>
    <range>6300</range>
    <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
  </LookAt>
  <LineString>
    <extrude>1</extrude>
    <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
    <coordinates>
      146.825,12.233,400
      146.820,12.222,400
      146.812,12.212,400
      146.796,12.209,400
      146.788,12.205,400
    </coordinates>
  </LineString>
</Placemark>

</kml>
```

<gx:AnimatedUpdate>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:AnimatedUpdate>
  <gx:duration>0.0</gx:duration>     <!-- double, specifies time in seconds -->
  <Update>
    <targetHref>...</targetHref>     <!-- required; can contain a URL or be left
blank -->
                                      <!-- (to target elements within the same
file) -->
    <Change>...</Change>
    <Create>...</Create>
    <Delete>...</Delete>
  </Update>
  <gx:delayedStart>0</gx:delayedStart>  <!-- double, specifies time in seconds --
>
</gx:AnimatedUpdate>
```

Description

`<gx:AnimatedUpdate>` controls changes during a tour to KML features, using `<Update>`. Changes to KML features will not modify the DOM - that is, any changes will be reverted when the tour is over, and will not be saved in the KML at any time.

`<gx:AnimatedUpdate>` should also contain a `<gx:duration>` value to specify the length of time in seconds over which the update takes place. Integer, float, and color fields are smoothly animated from original to new value across the duration; boolean, string, and other values that don't lend to interpolation are updated at the end of the duration.

Refer to Tour timelines in the **Touring** chapter of the **KML Developer's Guide** for information about `<gx:AnimatedUpdate>` and the tour timeline.

<gx:duration>

Specifies the length of time, in seconds, over which the update takes place.

<gx:delayedStart>

Specifies the number of seconds to wait (after the inline start position) before starting the update.

Example

The example below demonstrates a change in icon size. This change will be animated over a 5-second duration.

animatedupdate_example.kml

```
<?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">

<Document>
  <name>gx:AnimatedUpdate example</name>

  <Style id="pushpin">
    <IconStyle id="mystyle">
      <Icon>
        <href>http://maps.google.com/mapfiles/kml/pushpin/ylw-pushpin.png</href>
        <scale>1.0</scale>
      </Icon>
    </IconStyle>
  </Style>

  <Placemark id="mountainpin1">
    <name>Pin on a mountaintop</name>
    <styleUrl>#pushpin</styleUrl>
    <Point>
      <coordinates>170.1435558771009,-43.60505741890396,0</coordinates>
    </Point>
  </Placemark>

  <gx:Tour>
    <name>Play me!</name>

    <gx:Playlist>

      <gx:FlyTo>
        <gx:flyToMode>bounce</gx:flyToMode>
        <gx:duration>3</gx:duration>
        <Camera>
          <longitude>170.157</longitude>
          <latitude>-43.671</latitude>
          <altitude>9700</altitude>
          <heading>-6.333</heading>
          <tilt>33.5</tilt>
        </Camera>
      </gx:FlyTo>

      <gx:AnimatedUpdate>
        <gx:duration>5</gx:duration>
        <Update>
          <targetHref></targetHref>
          <Change>
            <IconStyle targetId="mystyle">
              <scale>10.0</scale>
            </IconStyle>
          </Change>
        </Update>
      </gx:AnimatedUpdate>

      <gx:Wait>
        <gx:duration>5</gx:duration>
      </gx:Wait>

    </gx:Playlist>
  </gx:Tour>

</Document>
</kml>
```

Extends

- <gx:TourPrimitive>

Contains

- <gx:duration>
- <Update>

<BalloonStyle>

Syntax

```
<BalloonStyle id="ID">
  <!-- specific to BalloonStyle -->
  <bgColor>ffffffff</bgColor>            <!-- kml:color -->
  <textColor>ff000000</textColor>        <!-- kml:color -->
  <text>...</text>                       <!-- string -->
  <displayMode>default</displayMode>     <!-- kml:displayModeEnum -->
</BalloonStyle>
```

Description

Specifies how the description balloon for placemarks is drawn. The <bgColor>, if specified, is used as the background color of the balloon. See *<Feature>* for a diagram illustrating how the default description balloon appears in Google Earth.

Elements Specific to BalloonStyle

<bgColor>

Background color of the balloon (optional). Color and opacity (alpha) values are expressed in hexadecimal notation. The range of values for any one color is 0 to 255 (`00` to `ff`). The order of expression is *aabbggrr,* where *aa=alpha* (00 to ff); *bb=blue* (00 to ff); *gg=green* (00 to ff); *rr=red* (00 to ff). For alpha, `00` is fully transparent and `ff` is fully opaque. For example, if you want to apply a blue color with 50 percent opacity to an overlay, you would specify the following: `<bgColor>7fff0000</bgColor>`, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00. The default is opaque white (ffffffff).

Note: The use of the <color> element within <BalloonStyle> has been deprecated. Use <bgColor> instead.

<textColor>

Foreground color for text. The default is black (ff000000).

<text>

Text displayed in the balloon. If no text is specified, Google Earth draws the default balloon (with the Feature <name> in boldface, the Feature <description>, links for driving directions, a white background, and a tail that is attached to the point coordinates of the Feature, if specified).

You can add entities to the <text> tag using the following format to refer to a child element of Feature: **$[name]**, **$[description]**, **$[address]**, **$[id]**, **$[Snippet]**. Google Earth looks in the current Feature for the corresponding string entity and substitutes that information in the balloon. To include *To here - From here* driving directions in the balloon, use the **$[geDirections]** tag. To prevent the driving directions links from appearing in a balloon, include the <text> element with some content, or with $[description] to substitute the basic Feature <description>.

For example, in the following KML excerpt, **$[name]** and **$[description]** fields will be replaced by the <name> and <description>

fields found in the Feature elements that use this BalloonStyle:

```
<text>This is $[name], whose description is:<br/>$[description]</text>
```

<displayMode>

If <displayMode> is *default*, Google Earth uses the information supplied in <text> to create a balloon . If <displayMode> is *hide*, Google Earth does not display the balloon. In Google Earth, clicking the List View icon for a Placemark whose balloon's <displayMode> is *hide* causes Google Earth to fly to the Placemark.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>BalloonStyle.kml</name>
  <open>1</open>
  <Style id="exampleBalloonStyle">
    <BalloonStyle>
      <!-- a background color for the balloon -->
      <bgColor>ffffffbb</bgColor>
      <!-- styling of the balloon text -->
      <text><![CDATA[
      <b><font color="#CC0000" size="+3">$[name]</font></b>
      <br/><br/>
      <font face="Courier">$[description]</font>
      <br/><br/>
      Extra text that will appear in the description balloon
      <br/><br/>
      <!-- insert the to/from hyperlinks -->
      $[geDirections]
      ]]></text>
    </BalloonStyle>
  </Style>
  <Placemark>
    <name>BalloonStyle</name>
    <description>An example of BalloonStyle</description>
    <styleUrl>#exampleBalloonStyle</styleUrl>
    <Point>
      <coordinates>-122.370533,37.823842,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Extends

- *<ColorStyle>*

Contained By

- <Style>

<gx:balloonVisibility>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:balloonVisibility>0</gx:balloonVisibility>    <!-- 0 (not visible) or 1
(visible) -->
```

Description

Toggles visibility of a description balloon. The balloon to be updated must be identified by the object's XML ID (e.g. `<Placemark targetId="xxx">`).

Examples

The first example shows `<gx:balloonVisibility>` with a Placemark. When the placemark is loaded, the description balloon is opened.

balloonvisibility_example.kml

```
<?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">

  <Placemark>
    <name>Eiffel Tower</name>
    <description>
        Located in Paris, France.

        This description balloon opens
        when the Placemark is loaded.
    </description>
    <gx:balloonVisibility>1</gx:balloonVisibility>
    <Point>
      <coordinates>2.294785,48.858093,0</coordinates>
    </Point>
  </Placemark>

</kml>
```

The second example shows the use of `<gx:balloonVisibility>` within a tour. A number of balloons are opened and closed during the tour, providing information to the viewer.

balloonvisibility_tourexample.kml

```
<?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">

  <Document>
    <name>balloonVisibility Example</name>
    <open>1</open>

    <gx:Tour>
      <name>Play me</name>
      <gx:Playlist>

        <gx:FlyTo>
          <gx:duration>8.0</gx:duration>
          <gx:flyToMode>bounce</gx:flyToMode>
          <LookAt>
            <longitude>-119.748584</longitude>
            <latitude>33.736266</latitude>
            <altitude>0</altitude>
            <heading>-9.295926</heading>
            <tilt>84.0957450</tilt>
            <range>4469.850414</range>
            <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
          </LookAt>
        </gx:FlyTo>

        <gx:AnimatedUpdate>
          <gx:duration>0.0</gx:duration>
          <Update>
            <targetHref/>
            <Change>
              <Placemark targetId="underwater1">
                <gx:balloonVisibility>1</gx:balloonVisibility>
              </Placemark>
            </Change>
          </Update>
        </gx:AnimatedUpdate>

        <gx:Wait>
          <gx:duration>4.0</gx:duration>
        </gx:Wait>

        <gx:AnimatedUpdate>
          <gx:duration>0.0</gx:duration>
          <Update>
            <targetHref/>
            <Change>
              <Placemark targetId="underwater1">
                <gx:balloonVisibility>0</gx:balloonVisibility>
              </Placemark>
            </Change>
          </Update>
        </gx:AnimatedUpdate>

        <gx:FlyTo>
          <gx:duration>3</gx:duration>
          <gx:flyToMode>smooth</gx:flyToMode>
          <LookAt>
            <longitude>-119.782630</longitude>
            <latitude>33.862855</latitude>
            <altitude>0</altitude>
            <heading>-9.314858</heading>
            <tilt>84.117317</tilt>
            <range>6792.665540</range>
            <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
          </LookAt>
        </gx:FlyTo>

        <gx:AnimatedUpdate>
          <gx:duration>0.0</gx:duration>
          <Update>
            <targetHref/>
            <Change>
              <Placemark targetId="underwater2">
                <gx:balloonVisibility>1</gx:balloonVisibility>
              </Placemark>
            </Change>
          </Update>
        </gx:AnimatedUpdate>

        <gx:Wait>
          <gx:duration>4.0</gx:duration>
        </gx:Wait>

        <gx:AnimatedUpdate>
          <gx:duration>0.0</gx:duration>
          <Update>
            <targetHref/>
            <Change>
              <Placemark targetId="underwater2">
                <gx:balloonVisibility>0</gx:balloonVisibility>
              </Placemark>
            </Change>
          </Update>
        </gx:AnimatedUpdate>

        <gx:FlyTo>
          <gx:duration>3</gx:duration>
          <gx:flyToMode>smooth</gx:flyToMode>
          <LookAt>
            <longitude>-119.849578</longitude>
            <latitude>33.968515</latitude>
            <altitude>0</altitude>
            <heading>-173.948935</heading>
            <tilt>23.063392</tilt>
            <range>3733.666023</range>
            <altitudeMode>relativeToGround</altitudeMode>
          </LookAt>
        </gx:FlyTo>

        <gx:AnimatedUpdate>
          <gx:duration>0.0</gx:duration>
          <Update>
            <targetHref/>
            <Change>
              <Placemark targetId="onland">
                <gx:balloonVisibility>1</gx:balloonVisibility>
              </Placemark>
            </Change>
          </Update>
        </gx:AnimatedUpdate>

        <gx:Wait>
          <gx:duration>4.0</gx:duration>
        </gx:Wait>

      </gx:Playlist>
    </gx:Tour>

    <Placemark id="underwater1">
      <name>Underwater off the California Coast</name>
      <description>
        The tour begins near the Santa Cruz Canyon,
        off the coast of California, USA.
      </description>
      <Point>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>-119.749531,33.715059,0</coordinates>
      </Point>
    </Placemark>

    <Placemark id="underwater2">
      <name>Still swimming...</name>
      <description>We're about to leave the ocean, and visit the
coast...</description>
      <Point>
        <gx:altitudeMode>clampToSeaFloor</gx:altitudeMode>
        <coordinates>-119.779550,33.829268,0</coordinates>
      </Point>
    </Placemark>

    <Placemark id="onland">
      <name>The end</name>
      <description>
```

```
        <![CDATA[The end of our simple tour.
        Use <gx:balloonVisibility>1</gx:balloonVisibility>
        to show description balloons.]]>
      </description>
      <Point>
        <coordinates>-119.849578,33.968515,0</coordinates>
      </Point>
    </Placemark>


  </Document>
</kml>
```

Extends

- <Feature>

<Camera>

Syntax

```
<Camera id="ID">
  <!-- inherited from AbstractView element -->
  <TimePrimitive>...</TimePrimitive>  <!-- gx:TimeSpan or gx:TimeStamp -->
  <gx:ViewerOptions>
    <option> name=" " type="boolean">     <!--
name="streetview", "historicalimagery", "sunlight", or "groundnavigation" -->
    </option>
  </gx:ViewerOptions>

  <!-- specific to Camera -->
  <longitude>0</longitude>            <!-- kml:angle180 -->
  <latitude>0</latitude>              <!-- kml:angle90 -->
  <altitude>0</altitude>              <!-- double -->
  <heading>0</heading>                <!-- kml:angle360 -->
  <tilt>0</tilt>                      <!-- kml:anglepos180 -->
  <roll>0</roll>                      <!-- kml:angle180 -->
  <altitudeMode>clampToGround</altitudeMode>
        <!-- kml:altitudeModeEnum: relativeToGround, clampToGround, or absolute -
->
        <!-- or, gx:altitudeMode can be substituted: clampToSeaFloor,
relativeToSeaFloor -->
</Camera>
```

Description

Defines the virtual camera that views the scene. This element defines the ***position*** of the camera relative to the Earth's surface as well as the *viewing direction* of the camera. The camera *position* is defined by <longitude>, <latitude>, <altitude>, and either <altitudeMode> or <gx:altitudeMode>. The *viewing direction* of the camera is defined by <heading>, <tilt>, and <roll>. <Camera> can be a child element of any Feature or of <NetworkLinkControl>. A parent element cannot contain both a <Camera> and a <LookAt> at the same time.

<Camera> provides full six-degrees-of-freedom control over the view, so you can position the Camera in space and then rotate it around the *X*, *Y*, and *Z* axes. Most importantly, you can tilt the camera view so that you're looking above the horizon into the sky.

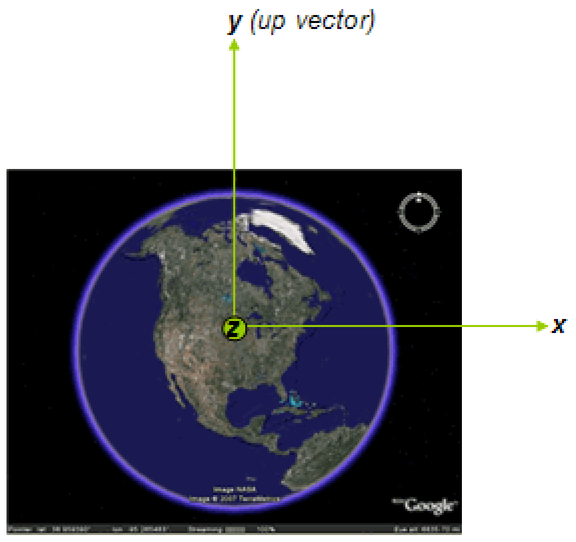
<Camera> can also contain a TimePrimitive (<gx:TimeSpan> or <gx:TimeStamp>). Time values in Camera affect historical imagery, sunlight, and the display of time-stamped features. For more information, read Time with AbstractViews in the **Time and Animation** chapter of the Developer's Guide.

Defining a View

Within a Feature or <NetworkLinkControl>, use either a <Camera> or a <LookAt> object (but not both in the same object). The <Camera> object defines the viewpoint in terms of the viewer's position and orientation. The <Camera> object allows you to specify a view that is not on the Earth's surface. The <LookAt> object defines the viewpoint in terms of what is being viewed. The <LookAt> object is more limited in scope than <Camera> and generally requires that the view direction intersect the Earth's surface.

The following diagram shows the *X*, *Y*, and *Z* axes, which are attached to the virtual camera.

- The *X* axis points toward the right of the camera and is called the *right vector*.
- The *Y* axis defines the "up" direction relative to the screen and is called the *up vector*.
- The *Z* axis points from the center of the screen toward the eye point. The camera looks down the −*Z* axis, which is called the *view vector*.



Order of Transformations

The order of rotation is important. By default, the camera is looking straight down the −Z axis toward the Earth. Before rotations are performed, the camera is translated along the *Z* axis to <altitude>. The order of transformations is as follows:

1. **<altitude>** - translate along the *Z* axis to <altitude>
2. **<heading>** - rotate around the *Z* axis.
3. **<tilt>** - rotate around the *X* axis.
4. **<roll>** - rotate around the *Z* axis (again).

Note that each time a rotation is applied, two of the camera axes change their orientation.

Elements Specific to Camera

<longitude>

Longitude of the virtual camera (eye point). Angular distance in degrees, relative to the Prime Meridian. Values west of the Meridian range from −180 to 0 degrees. Values east of the Meridian range from 0 to 180 degrees.

<latitude>

Latitude of the virtual camera. Degrees north or south of the Equator (0 degrees). Values range from −90 degrees to 90 degrees.

<altitude>

Distance of the camera from the earth's surface, in meters. Interpreted according to the Camera's <altitudeMode> or <gx:altitudeMode>.

<heading>

Direction (azimuth) of the camera, in degrees. Default=0 (true North). (See diagram.) Values range from 0 to 360 degrees.

<tilt>

Rotation, in degrees, of the camera around the *X* axis. A value of 0 indicates that the view is aimed straight down toward the earth (the most common case). A value for 90 for <tilt> indicates that the view is aimed toward the horizon. Values greater than 90 indicate that the view is pointed up into the sky. Values for <tilt> are clamped at +180 degrees.

<roll>

Rotation, in degrees, of the camera around the ***Z*** axis. Values range from −180 to +180 degrees.

<altitudeMode>

Specifies how the <altitude> specified for the Camera is interpreted. Possible values are as follows:

- **relativeToGround** - (default) Interprets the <altitude> as a value in meters above the ground. If the point is over water, the

<altitude> will be interpreted as a value in meters above sea level. See <gx:altitudeMode> below to specify points relative to the sea floor.

- **clampToGround** - For a camera, this setting also places the camera **relativeToGround**, since putting the camera exactly at terrain height would mean that the eye would intersect the terrain (and the view would be blocked).
- **absolute** - Interprets the <altitude> as a value in meters above sea level.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the Camera will be positioned on the sea floor. If the point is on land rather than at sea, the Camera will be positioned on the ground.

Extends

- *<AbstractView>*

Contained By

- Any element derived from *<Feature>*
- <NetworkLinkControl>

<ColorStyle>

Syntax

```
<!-- abstract element; do not create -->
<!-- ColorStyle id="ID" -->          <!--
IconStyle,LabelStyle,LineStyle,PolyStyle -->
  <color>ffffffff</color>            <!-- kml:color -->
  <colorMode>normal</colorMode>      <!-- kml:colorModeEnum: normal or random -->
<!-- /ColorStyle -->
```

Description

This is an abstract element and cannot be used directly in a KML file. It provides elements for specifying the color and color mode of extended style types.

Elements Specific to ColorStyle

<color>

Color and opacity (alpha) values are expressed in hexadecimal notation. The range of values for any one color is 0 to 255 (`00` to `ff`). For alpha, `00` is fully transparent and `ff` is fully opaque. The order of expression is *aabbggrr,* where *aa=alpha* (00 to ff); *bb=blue* (00 to ff); *gg=green* (00 to ff); *rr=red* (00 to ff). For example, if you want to apply a blue color with 50 percent opacity to an overlay, you would specify the following: `<color>7fff0000</color>`, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00.

<colorMode>

Values for <colorMode> are **normal** (no effect) and **random**. A value of **random** applies a random linear scale to the base <color> as follows.

- To achieve a truly random selection of colors, specify a base <color> of white (ffffffff).
- If you specify a single color component (for example, a value of ff0000ff for *red*), random color values for that one component (red) will be selected. In this case, the values would range from 00 (*black*) to ff (*full red*).
- If you specify values for two or for all three color components, a random linear scale is applied to each color component, with results ranging from black to the maximum values specified for each component.
- The opacity of a color comes from the alpha component of <color> and is never randomized.

Extends

- *<Object>*

Extended By

- <IconStyle>
- <LabelStyle>
- <LineStyle>
- <PolyStyle>

<Container>

Syntax

```
<!-- abstract element; do not create -->
<!-- Container id="ID" -->                <!-- Document,Folder -->
  <!-- inherited from Feature element -->
  <name>...</name>                        <!-- string -->
  <visibility>1</visibility>              <!-- boolean -->
  <open>0</open>                          <!-- boolean -->
  <address>...</address>                  <!-- string -->
  <AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </AddressDetails>                   <!-- string -->
  <phoneNumber>...</phoneNumber>          <!-- string -->
  <Snippet maxLines="2">...</Snippet>     <!-- string -->
  <description>...</description>          <!-- string -->
  <AbstractView>...</AbstractView>        <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>                <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>
  <atom:author>...<atom:author>   <!-- xmlns:atom="http://www.w3.org/2005/Atom" --
>
  <atom:link href=" "/>

  <!-- specific to Container -->
  <!-- 0 or more Features -->
<!-- /Container -->
```

Description

This is an abstract element and cannot be used directly in a KML file. A Container element holds one or more Features and allows the creation of nested hierarchies.

Extends

- *<Feature>*

Extended By

- <Document>
- <Folder>

<Document>

Syntax

```
<Document id="ID">
  <!-- inherited from Feature element -->
  <name>...</name>                        <!-- string -->
  <visibility>1</visibility>              <!-- boolean -->
  <open>0</open>                          <!-- boolean -->
  <atom:author>...<atom:author>           <!-- xmlns:atom -->
  <atom:link href=" "/>                   <!-- xmlns:atom -->
  <address>...</address>                  <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>          <!-- string -->
  <Snippet maxLines="2">...</Snippet>     <!-- string -->
  <description>...</description>          <!-- string -->
  <AbstractView>...</AbstractView>        <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>                <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- specific to Document -->
  <!-- 0 or more Schema elements -->
  <!-- 0 or more Feature elements -->
</Document>
```

Description

A Document is a container for features and styles. This element is required if your KML file uses *shared styles*. It is recommended that you use shared styles, which require the following steps:

1. Define all Styles in a Document. Assign a unique **ID** to each Style.
2. Within a given Feature or StyleMap, reference the Style's ID using a <styleUrl> element.

Note that shared styles are not inherited by the Features in the Document.

Each Feature must explicitly reference the styles it uses in a <styleUrl> element. For a Style that applies to a Document (such as ListStyle), the Document itself must explicitly reference the <styleUrl>. For example:

```
<Document>
  <Style id="myPrettyDocument">
   <ListStyle> ... </ListStyle>

  </Style>
  <styleUrl#myPrettyDocument">
  ...
</Document>
```

Do not put shared styles within a Folder.

The following example illustrates use of a shared style.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>Document.kml</name>
  <open>1</open>
  <Style id="exampleStyleDocument">
    <LabelStyle>
      <color>ff0000cc</color>
    </LabelStyle>
  </Style>
  <Placemark>
    <name>Document Feature 1</name>
    <styleUrl>#exampleStyleDocument</styleUrl>
    <Point>
      <coordinates>-122.371,37.816,0</coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name>Document Feature 2</name>
    <styleUrl>#exampleStyleDocument</styleUrl>
    <Point>
      <coordinates>-122.370,37.817,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Extends

- *<Container>*

Contains

- 0 or more elements derived from *<Feature>*
- 0 or more elements derived from *<StyleSelector>*
- 0 or more elements derived from <Schema>

<gx:duration>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:duration>0.0</gx:duration>            <!-- double -->
```

Description

`<gx:duration>` extends `gx:TourPrimitive` by specifying a time-span for events. The time is written as seconds using XML's double datatype.

Duration and <gx:FlyTo>

When a duration is included within a `<gx:FlyTo>` element, it specifies the length of time that the browser takes to fly from the previous point to the specified point.

```
<gx:FlyTo>
  <gx:flyToMode>bounce</gx:flyToMode>
  <gx:duration>10.2</gx:duration>

  <!-- AbstractView -->
    ...
  <!-- /AbstractView -->
</gx:FlyTo>
```

Duration and <gx:AnimatedUpdate>

Specifies the length of time over which the update takes place. Integer, float, and color fields are smoothly animated from original to new value across the duration; boolean, string, and other values that don't lend to interpolation are updated at the end of the duration.

```
<gx:AnimatedUpdate>
  <gx:duration>5.0</gx:duration>
  <Update>

    ....

  </Update>
</gx:AnimatedUpdate>
```

<ExtendedData>

Syntax

```
<ExtendedData>
  <Data name="string">
    <displayName>...</displayName>    <!-- string -->
    <value>...</value>                <!-- string -->
  </Data>
  <SchemaData schemaUrl="anyURI">
    <SimpleData name=""> ... </SimpleData>   <!-- string -->
  </SchemaData>
  <namespace_prefix:other>...</namespace_prefix:other>
</ExtendedData>
```

Description

The ExtendedData element offers three techniques for adding custom data to a KML Feature (NetworkLink, Placemark, GroundOverlay, PhotoOverlay, ScreenOverlay, Document, Folder). These techniques are

- Adding untyped data/value pairs using the <Data> element (basic)
- Declaring new typed fields using the <Schema> element and then instancing them using the <SchemaData> element (advanced)
- Referring to XML elements defined in other namespaces by referencing the external namespace within the KML file (basic)

These techniques can be combined within a single KML file or Feature for different pieces of data.

For more information, see Adding Custom Data in "Topics in KML."

Elements Specific to ExtendedData

<Data name ="*string*">

Creates an untyped *name/value* pair. The name can have two versions: *name* and *displayName*. The *name* attribute is used to identify the data pair within the KML file. The *displayName* element is used when a properly formatted name, with spaces and HTML formatting, is displayed in Google Earth. In the <text> element of <BalloonStyle>, the notation **$[name/displayName]** is replaced with <displayName>. If you substitute the value of the *name* attribute of the <Data> element in this format (for example, $[holeYardage], the attribute value is replaced with <value>. By default, the Placemark's balloon displays the name/value pairs associated with it.

<displayName>

An optional formatted version of *name*, to be used for display purposes.

<value>

Value of the data pair.

```
<Placemark>
  <name>Club house</name>
  <ExtendedData>
    <Data name="holeNumber">
      <value>1</value>
    </Data>
    <Data name="holeYardage">
      <value>234</value>
    </Data>
    <Data name="holePar">
      <value>4</value>
    </Data>
  </ExtendedData>
</Placemark>
```

<SchemaData schemaUrl="*anyURI*">

This element is used in conjunction with <Schema> to add typed custom data to a KML Feature. The Schema element (identified by the *schemaUrl* attribute) declares the custom data type. The actual data objects ("instances" of the custom data) are defined using the SchemaData element.

The <schemaURL> can be a full URL, a reference to a Schema ID defined in an external KML file, or a reference to a Schema ID defined in the same KML file. All of the following specifications are acceptable:

```
schemaUrl="http://host.com/PlacesIHaveLived.kml#my-schema-id"
```

```
schemaUrl="AnotherFile.kml#my-schema-id"
```

```
schemaUrl="#schema-id"   <!-- same file -->
```

The Schema element is always a child of Document. The ExtendedData element is a child of the Feature that contains the custom data.

<SimpleData name="*string*">

This element assigns a value to the custom data field identified by the *name* attribute. The type and name of this custom data field are declared in the <Schema> element.

Here is an example of defining two custom data elements:

```
<Placemark>
  <name>Easy trail</name>
  <ExtendedData>
    <SchemaData schemaUrl="#TrailHeadTypeId">
      <SimpleData name="TrailHeadName">Pi in the sky</SimpleData>
      <SimpleData name="TrailLength">3.14159</SimpleData>
      <SimpleData name="ElevationGain">10</SimpleData>
    </SchemaData>
    </ExtendedData>
    <Point>
      <coordinates>-122.000,37.002</coordinates>
    </Point>
</Placemark>
<Placemark>
  <name>Difficult trail</name>
  <ExtendedData>
    <SchemaData schemaUrl="#TrailHeadTypeId">
      <SimpleData name="TrailHeadName">Mount Everest</SimpleData>
      <SimpleData name="TrailLength">347.45</SimpleData>
      <SimpleData name="ElevationGain">10000</SimpleData>
    </SchemaData>
  </ExtendedData>
  <Point>
    <coordinates>-122.000,37.002</coordinates>
  </Point>
</Placemark>
```

<namespace_prefix:other>

This element allows you to add untyped custom data. Be sure to reference the namespace prefix in the <kml> element of your file or as an attribute of the <ExtendedData> element and to prefix the name of each data element with the namespace prefix. Custom data added in this manner is preserved in the KML file but is not used by Google Earth in any way. It is always saved along with the file.

The following example shows using the "camp" namespace prefix:

```
<ExtendedData xmlns:prefix="camp">
  <camp:number>14</camp:number>
  <camp:parkingSpaces>2</camp:parkingSpaces>
  <camp:tentSites>4</camp:tentSites>
</ExtendedData>
```

Contained By

- Any element derived from *<Feature>*

See Also

- Schema

<Feature>

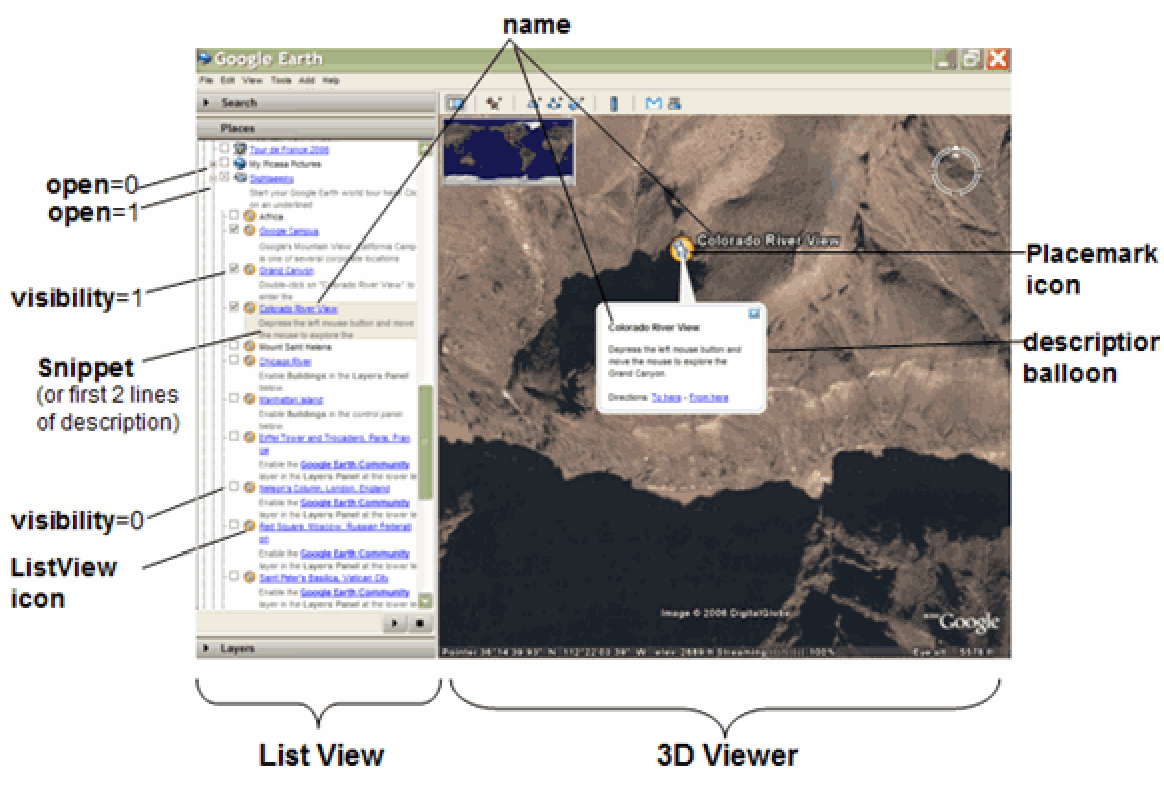
Syntax

```
<!-- abstract element; do not create -->
<!-- Feature id="ID" -->                <!-- Document,Folder,
                                             NetworkLink,Placemark,

GroundOverlay,PhotoOverlay,ScreenOverlay -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>            <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->
<-- /Feature -->
```

Description

This is an abstract element and cannot be used directly in a KML file. The following diagram shows how some of a Feature's elements appear in Google Earth.



Elements Specific to Feature

<name>

User-defined text displayed in the 3D viewer as the label for the object (for example, for a Placemark, Folder, or NetworkLink).

<visibility>

Boolean value. Specifies whether the feature is drawn in the 3D viewer when it is initially loaded. In order for a feature to be visible, the <visibility> tag of all its ancestors must also be set to 1. In the Google Earth List View, each Feature has a checkbox that allows the user to control visibility of the Feature.

<open>

Boolean value. Specifies whether a Document or Folder appears closed or open when first loaded into the Places panel. 0=collapsed (the default), 1=expanded. See also <ListStyle>. This element applies only to Document, Folder, and NetworkLink.

<atom:author>

KML 2.2 supports new elements for including data about the author and related website in your KML file. This information is displayed in geo search results, both in Earth browsers such as Google Earth, and in other applications such as Google Maps. The ascription elements used in KML are as follows:

- **atom:author** element - parent element for atom:name
- **atom:name** element - the name of the author
- **atom:link** element - contains the *href* attribute
- **href** attribute - URL of the web page containing the KML/KMZ file

These elements are defined in the Atom Syndication Format. The complete specification is found at http://atompub.org. (see the sample that follows).

The <atom:author> element is the parent element for <atom:name>, which specifies the author of the KML feature.

<atom:link href="..." >

Specifies the URL of the website containing this KML or KMZ file. Be sure to include the namespace for this element in any KML file that uses it: `xmlns:atom="http://www.w3.org/2005/Atom"` (see the sample that follows).

<address>

A string value representing an unstructured address written as a standard street, city, state address, and/or as a postal code. You can use the <address> tag to specify the location of a point instead of using latitude and longitude coordinates. (However, if a <Point> is provided, it takes precedence over the <address>.) To find out which locales are supported for this tag in Google Earth, go to the Google Maps Help.

<xal:AddressDetails>

A structured address, formatted as xAL, or eXtensible Address Language, an international standard for address formatting. <xal:AddressDetails> is used by KML for geocoding in Google Maps only. For details, see the Google Maps API documentation. Currently, Google Earth does not use this element; use <address> instead. Be sure to include the namespace for this element in any KML file that uses it: `xmlns:xal="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"`

<phoneNumber>

A string value representing a telephone number. This element is used by Google Maps Mobile only. The industry standard for Java-enabled cellular phones is RFC2806.
For more information, see http://www.ietf.org/rfc /rfc2806.txt.

<Snippet maxLines="2" >

A short description of the feature. In Google Earth, this description is displayed in the Places panel under the name of the feature. If a Snippet is not supplied, the first two lines of the <description> are used. In Google Earth, if a Placemark contains both a description and a Snippet, the <Snippet> appears beneath the Placemark in the Places panel, and the <description> appears in the Placemark's description balloon. This tag does not support HTML markup. <Snippet> has a **maxLines** attribute, an integer that specifies the maximum number of lines to display.

<description>

User-supplied content that appears in the description balloon.

The supported content for the <description> element changed from Google Earth 4.3 to 5.0. Specific information for each version is listed out below, followed by information common to both.

Google Earth 5.0

Google Earth 5.0 (and later) supports plain text content, as well as full HTML and JavaScript, within description balloons. Contents of the description tag are rendered by the WebKit open source web browser engine, and are displayed as they would be in any WebKit-based browser.

General restrictions

Links to local files are generally not allowed. This prevents malicious code from damaging your system or accessing your data. Should you wish to allow access to your local filesystem, select **Preferences** > **Allow access to local files and personal data**. Links to image files on the local filesystem are always allowed, if contained within an <img> tag.

Content that has been compressed into a KMZ file can be accessed, even if on the local filesystem.

Cookies are enabled, but for the purposes of the same-origin policy, local content does not share a domain with any other content (including other local content).

HTML

HTML is mostly rendered as it would be in any WebKit browser.

Targets are ignored when included in HTML written directly into the KML; all such links are opened as if the target is set to _blank. Any specified targets are ignored.

HTML that is contained in an iFrame, however, or dynamically generated with JavaScript or DHTML, will use target="_self" as the default. Other targets can be specified and are supported.

The contents of KMZ files, local anchor links, and `;flyto` methods cannot be targeted from HTML contained within an iFrame.

If the user specifies `width="100%"` for the width of an iFrame, then the iFrame's width will be dependent on all the other content in the balloon—it should essentially be ignored while calculating layout size. This rule applies to any other block element inside the balloon as well.

JavaScript

Most JavaScript is supported. Dialog boxes can not be created - functions such as alert() and prompt() will not be displayed. They will, however, be written to the system console, as will other errors and exceptions.

CSS

CSS is allowed. As with CSS in a regular web browser, CSS can be used to style text, page elements, and to control the size and appearance of the description balloon.

Google Earth 4.3

The <description> element supports plain text as well as a subset of HTML formatting elements, including tables (see KML example below). It does not support other web-based technology, such as dynamic page markup (PHP, JSP, ASP), scripting languages (VBScript, Javascript), nor application languages (Java, Python). In Google Earth release 4.2, video is supported. (See Example below.)

Common information

If your description contains no HTML markup, Google Earth attempts to format it, replacing newlines with <br> and wrapping URLs with anchor tags. A valid URL string for the World Wide Web is automatically converted to a hyperlink to that URL (e.g., *http://www.google.com*). Consequently, you do not need to surround a URL with the `<a href="http://..."></a>` tags in order to achieve a simple link.

When using HTML to create a hyperlink around a specific word, or when including images in the HTML, you must use HTML entity references or the CDATA element to escape angle brackets, apostrophes, and other special characters. The CDATA element tells the XML parser to ignore special characters used within the brackets. This element takes the form of:

```
<![CDATA[
  special characters here
]]>
```

If you prefer not to use the CDATA element, you can use entity references to replace all the special characters.

```
<description>
  <![CDATA[
This is an image
<img src="icon.jpg">
and we have a link http://www.google.com.
  ]]>
</description>
```

Other Behavior Specified Through Use of the <a> Element

KML supports the use of two attributes within the <a> element: **href** and **type**.

The anchor element <a> contains an *href* attribute that specifies a URL.

If the *href* is a KML file and has a *.kml* or *.kmz* file extension, Google Earth loads that file directly when the user clicks it. If the URL ends with an extension not known to Google Earth (for example, *.html*), the URL is sent to the browser.

The *href* can be a fragment URL (that is, a URL with a # sign followed by a KML identifier). When the user clicks a link that includes a fragment URL, by default the browser flies to the Feature whose ID matches the fragment. If the Feature has a LookAt or Camera element, the Feature is viewed from the specified viewpoint.

The behavior can be further specified by appending one of the following three strings to the fragment URL:

- *;flyto* (default) - fly to the Feature
- *;balloon* - open the Feature's balloon but do not fly to the Feature
- *;balloonFlyto* - open the Feature's balloon and fly to the Feature

For example, the following code indicates to open the file *CraftsFairs.kml*, fly to the Placemark whose ID is "Albuquerque," and open its balloon:

```
<description>
  <![CDATA[
    <a href="http://myServer.com/CraftsFairs.kml#Albuquerque;balloonFlyto">
      One of the Best Art Shows in the West</a>
  ]]>
</description>
```

The *type* attribute is used within the <a> element when the *href* does not end in *.kml* or *.kmz*, but the reference needs to be interpreted in the context of KML. Specify the following:

```
type="application/vnd.google-earth.kml+xml"
```

For example, the following URL uses the `type` attribute to notify Google Earth that it should attempt to load the file, even though the file extension is *.php*:

```
<a href="myserver.com/cgi-bin/generate-kml.php#placemark123"
   type="application/vnd.google-earth.kml+xml">
```

<AbstractView>

Defines a viewpoint associated with any element derived from Feature. See <Camera> and <LookAt>.

<TimePrimitive>

Associates this Feature with a period of time (<TimeSpan>) or a point in time (<TimeStamp>).

<styleUrl>

URL of a <Style> or <StyleMap> defined in a Document. If the style is in the same file, use a # reference. If the style is defined in an external file, use a full URL along with # referencing. Examples are

```
<styleUrl>#myIconStyleID</styleUrl>
```

```
<styleUrl>http://someserver.com/somestylefile.xml#restaurant</styleUrl>
```

```
<styleUrl>eateries.kml#my-lunch-spot</styleUrl>
```

<*StyleSelector*>

One or more Styles and StyleMaps can be defined to customize the appearance of any element derived from Feature or of the Geometry in a Placemark. (See <BalloonStyle>, <ListStyle>, <StyleSelector>, and the styles derived from <ColorStyle>.) A style defined within a Feature is called an "inline style" and applies only to the Feature that contains it. A style defined as the child of a <Document> is called a "shared style." A shared style must have an *id* defined for it. This id is referenced by one or more Features within the <Document>. In cases where a style element is defined both in a shared style and in an inline style for a Feature—that is, a Folder, GroundOverlay, NetworkLink, Placemark, or ScreenOverlay—the value for the Feature's inline style takes precedence over the value for the shared style.

<Region>

Features and geometry associated with a Region are drawn only when the Region is active. See <Region>.

<Metadata> (deprecated in KML 2.2; use <ExtendedData> instead)

<ExtendedData>

Allows you to add custom data to a KML file. This data can be (1) data that references an external XML schema, (2) untyped data/value pairs, or (3) typed data. A given KML Feature can contain a combination of these types of custom data.

Sample Use of HTML Elements within a Description

This example illustrates the complete set of HTML elements supported by the <description> element **in Google Earth 4.3**. Google Earth 5.0 and later supports full HTML and JavaScript.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Placemark>
  <name>Feature.kml</name>
  <Snippet maxLines="4">
    The snippet is a way of
    providing an alternative
    description that will be
    shown in the List view.
  </Snippet>
  <description>
    <![CDATA[
      Styles: <i>Italics</i>, <b>Bold</b>, <u>Underlined</u>,
      <s>Strike Out</s>, subscript<sub>subscript</sub>,
      superscript<sup>superscript</sup>,
      <big>Big</big>, <small>Small</small>, <tt>Typewriter</tt>,
      <em>Emphasized</em>, <strong>Strong</strong>, <code>Code</code>
      <hr />
      Fonts:
      <font color="red">red by name</font>,
      <font color="#408010">leaf green by hexadecimal RGB</font>,
      <font size=1>size 1</font>, <font size=2>size 2</font>,
      <font size=3>size 3</font>, <font size=4>size 4</font>,
      <font size=5>size 5</font>, <font size=6>size 6</font>,
      <font size=7>size 7</font>,
      <font face=times>Times</font>,
      <font face=verdana>Verdana</font>,
      <font face=arial>Arial</font>
      <br/>
      <hr />
      Links:
      <a href="http://doc.trolltech.com/3.3/qstylesheet.html">
      QT Rich Text Rendering
      </a>
      <br />
```

```
<hr />
Alignment:
<br />
<p align=left>left</p><p align=center>center</p>
<p align=right>right</p>
<hr />
Ordered Lists:
<br />
<ol><li>First</li><li>Second</li><li>Third</li></ol>
<ol type="a"><li>First</li><li>Second</li><li>Third</li></ol>
<ol type="A"><li>First</li><li>Second</li><li>Third</li></ol>
<hr />
Unordered Lists:
<br />
<ul><li>A</li><li>B</li><li>C</li></ul>
<ul type="circle"><li>A</li><li>B</li><li>C</li></ul>
<ul type="square"><li>A</li><li>B</li><li>C</li></ul>
<hr />
Definitions:
<br />
<dl>
<dt>Scrumpy</dt>
<dd>Hard English cider from the west country</dd>
<dt>Pentanque</dt>
<dd>A form of boules where the goal is to throw metal ball as
close as possible to a jack</dd>
</dl>
<hr />
Block Quote:
<br />
<blockquote>
We shall not cease from exploration<br />
And the end of all our exploring<br />
Will be to arrive where we started<br />
And know the place for the first time
</blockquote>
<br />
<hr />
Centered:
<br />
<center>See, I have a Rhyme assisting<br />
my feeble brain,<br />
its tasks oft-times resisting!</center>
<hr />
Headings:
<br />
<h1>Header 1</h1>
<h2>Header 2</h2>
<h3>Header 3</h3>
<h3>Header 4</h4>
<h3>Header 5</h5>
<hr />
Images:
<br />
<img src="http://earth.google.com/images/googleearth.gif" />
<br />
<i>Scaled image</i>
<br />
<img src="http://earth.google.com/images/googleearth.gif"
width="100" />
<br />
<hr />
Tables:
      <table border="1" padding="3" width="300">
      <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
      <tr><td>a</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>
      </table>
    ]]>
  </description>
  <Point>
    <coordinates>-122.378927,37.826793,0</coordinates>
  </Point>
</Placemark>
</kml>
```

Sample Use of Ascription Elements

This example shows use of the <atom:author>, <atom:name> and <atom:link> elements from the Atom namespace. Note that you need to reference this namespace within the <kml> element.

```
<kml xmlns="http://www.opengis.net/kml/2.2"
     xmlns:atom="http://www.w3.org/2005/Atom">
  <Document>
    <atom:author>
      <atom:name>J. K. Rowling</atom:name>
    </atom:author>
    <atom:link href="http://www.harrypotter.com" />
    <Placemark>
      <name>Hogwarts</name>
      <Point>
        <coordinates>1,1</coordinates>
      </Point>
    </Placemark>
    <Placemark>
      <name>Little Hangleton</name>
      <Point>
        <coordinates>1,2</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

This example shows how to embed a Flash video inside the Balloon.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Video Example</name>
    <Style id="sn_blue-dot_copy3">
      <IconStyle>
        <Icon>
          <href>http://www.google.com/intl/en_us/mapfiles/ms/icons/blue-
dot.png</href>
        </Icon>
      </IconStyle>
    </Style>
    <Placemark>
      <name>Placemark</name>
      <description>
        <![CDATA[<div style="font-size:larger">
          <div>
            <div style="width: 212px; font-size: 12px;">
              <b>The Spaghetti Film</b>
            </div>
```

```
              <div style="font-size: 11px;">
                <a target="_blank" href="http://www.youtube.com/?v=FICUvrVlyXc">
                  http://www.youtube.com/?v=FICUvrVlyXc</a><br>
              </div><br>
              <div style="margin-left: auto; margin-right:auto;">
                <object height="175" width="212">
                  <param value="http://www.youtube.com/v/FICUvrVlyXc" name="movie">
                  <param value="transparent" name="wmode">
                  <embed wmode="transparent" type="application/x-shockwave-flash"
                     src="http://www.youtube.com/v/FICUvrVlyXc" height="175"
                     width="212">
                 </object>
               </div>
             </div>
            </div>
            <div style="font-size: smaller; margin-top: 1em;">Saved from
              <a href="http://maps.google.com/ig/add?synd=mpl&pid=mpl&moduleurl=
                  http:%2F%2Fwww.google.com%2Fig%2Fmodules%2Fmapplets-
youtube.xml&hl=en&gl=us">
                  YouTubeVideos</a>
            </div>
          ]]>
      </description>
      <styleUrl>#sn_blue-dot_copy3</styleUrl>
      <Point>
        <coordinates>-93.47875999999999,45.083248,0</coordinates>
      </Point>
    </Placemark>
</Document>
</kml>
```

Extends

- <Object>

Extended By

- <Container>
- <Overlay>
- <Placemark>
- <NetworkLink>
- <gx:Tour>

<gx:FlyTo>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:FlyTo>
  <gx:duration>0.0</gx:duration>         <!-- double -->
  <gx:flyToMode>bounce</gx:duration>      <!-- smooth or bounce -->
  <!-- AbstractView -->                   <!-- Camera or LookAt -->
    ...
  <!-- /AbstractView -->
</gx:FlyTo>
```

Description

`<gx:FlyTo>` specifies a point in space to which the browser will fly during a tour. It must contain one AbstractView, and should contain `<gx:duration>` and `<gx:flyToMode>` elements, which specify the time it takes to fly to the defined point from the current point, and the method of flight, respectively.

<gx:flyToMode>

There are two allowed values for `<gx:flyToMode>`: **smooth**, and **bounce**.

- **Smooth** FlyTos allow for an unbroken flight from point to point to point (and on). An unbroken series of smooth FlyTos will begin and end at zero velocity, and will not slow at each point. A series of smooth FlyTos is broken by any of the following elements:
  - `<gx:flyToMode>bounce</gx:flyToMode>`
  - `<gx:Wait>`

  This means that velocity will be zero at the smooth FlyTo immediately preceding either of the above elements. A series of smooth FlyTos is not broken by <gx:AnimatedUpdate> elements.
- **Bounce** FlyTos each begin and end at zero velocity.

Example

```
<gx:FlyTo>
  <gx:duration>2.55</gx:duration>
  <gx:flyToMode>smooth</gx:flyToMode>
  <Camera>
    <longitude>-113.084448</longitude>
    <latitude>36.567081</latitude>
    <altitude>41277.571403</altitude>
    <heading>116.150227</heading>
    <altitudeMode>absolute</altitudeMode>
  </Camera>
</gx:FlyTo>
```

Extends

- <gx:TourPrimitive>

Contains

- <gx:duration>
- <gx:flyToMode>
- <AbstractView>

<Folder>

Syntax

```
<Folder id="ID">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>            <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>

  <!-- specific to Folder -->
  <!-- 0 or more Feature elements -->
```

```
</Folder>
```

Description

A Folder is used to arrange other Features hierarchically (Folders, Placemarks, NetworkLinks, or Overlays). A Feature is visible only if it and all its ancestors are visible.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Folder>
  <name>Folder.kml</name>
  <open>1</open>
  <description>
    A folder is a container that can hold multiple other objects
  </description>
  <Placemark>
    <name>Folder object 1 (Placemark)</name>
    <Point>
      <coordinates>-122.377588,37.830266,0</coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name>Folder object 2 (Polygon)</name>
    <Polygon>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            -122.377830,37.830445,0
            -122.377576,37.830631,0
            -122.377840,37.830642,0
            -122.377830,37.830445,0
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </Placemark>
  <Placemark>
    <name>Folder object 3 (Path)</name>
    <LineString>
      <tessellate>1</tessellate>
      <coordinates>
        -122.378009,37.830128,0 -122.377885,37.830379,0
      </coordinates>
    </LineString>
  </Placemark>
</Folder>
</kml>
```

Extends

- *<Container>*

Contains

- Any element derived from *<Feature>*

<Geometry>

Syntax

```
<!-- abstract element; do not create -->
```

```
<!-- Geometry id="ID" -->
                                             <!-- Point,LineString,LinearRing,
                                              Polygon,MultiGeometry,Model,
                                              gx:Track -->

<!-- /Geometry -->
```

Description

This is an abstract element and cannot be used directly in a KML file. It provides a placeholder object for all derived Geometry objects.

Extends

- *<Object>*

Extended By

- <Point>
- <LineString>
- <LinearRing>
- <Polygon>
- <MultiGeometry>
- <gx:MultiTrack>
- <Model>
- <gx:Track>

<GroundOverlay>

Syntax

```
<GroundOverlay id="ID">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>               <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- inherited from Overlay element -->
  <color>ffffffff</color>                   <!-- kml:color -->
  <drawOrder>0</drawOrder>                  <!-- int -->
  <Icon>...</Icon>

  <!-- specific to GroundOverlay -->
  <altitude>0</altitude>                    <!-- double -->
  <altitudeMode>clampToGround</altitudeMode>
     <!-- kml:altitudeModeEnum: clampToGround or absolute -->
            <!-- or, substitute gx:altitudeMode: clampToSeaFloor or
relativeToSeaFloor -->
  <LatLonBox>
    <north>...</north>                      <! kml:angle90 -->
    <south>...</south>                         <! kml:angle90 -->
    <east>...</east>                           <! kml:angle180 -->
    <west>...</west>                           <! kml:angle180 -->
    <rotation>0</rotation>                     <! kml:angle180 -->
  </LatLonBox>
  <gx:LatLonQuad>
    <coordinates>...</coordinates>             <!-- four lon,lat tuples -->
  </gx:LatLonQuad>
</GroundOverlay>
```

Description

This element draws an image overlay draped onto the terrain. The <href> child of <Icon> specifies the image to be used as the overlay. This file can be either on a local file system or on a web server. If this element is omitted or contains no <href>, a rectangle is drawn using the color and `LatLonBox` bounds defined by the ground overlay.

Elements Specific to GroundOverlay

<altitude>

Specifies the distance above the earth's surface, in meters, and is interpreted according to the altitude mode.

<altitudeMode>

Specifies how the <altitude>is interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore the altitude specification and drape the overlay over the terrain.
- **absolute** - Sets the altitude of the overlay relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of an overlay to 10 meters with an absolute altitude mode, the overlay will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the overlay will appear elevated above the terrain by 7 meters.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the overlay will be draped over the sea floor. If the point is on land rather than at sea, the overlay will be positioned on the ground.

<LatLonBox>

Specifies where the top, bottom, right, and left sides of a bounding box for the ground overlay are aligned.

- **<north>** Specifies the latitude of the north edge of the bounding box, in decimal degrees from 0 to ±90.
- **<south>** Specifies the latitude of the south edge of the bounding box, in decimal degrees from 0 to ±90.
- **<east>** Specifies the longitude of the east edge of the bounding box, in decimal degrees from 0 to ±180. (For overlays that overlap the meridian of 180° longitude, values can extend beyond that range.)
- **<west>** Specifies the longitude of the west edge of the bounding box, in decimal degrees from 0 to ±180. (For overlays that overlap the meridian of 180° longitude, values can extend beyond that range.)
- **<rotation>** Specifies a rotation of the overlay about its center, in degrees. Values can be ±180. The default is 0 (north). Rotations are specified in a counterclockwise direction.

```
<LatLonBox>
   <north>48.25475939255556</north>
   <south>48.25207367852141</south>
   <east>-90.86591508839973</east>
   <west>-90.8714285289695</west>
   <rotation>39.37878630116985</rotation>
</LatLonBox>
```

<gx:LatLonQuad>

Used for nonrectangular quadrilateral ground overlays.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<GroundOverlay>
   <name>GroundOverlay.kml</name>
   <color>7fffffff</color>
   <drawOrder>1</drawOrder>
   <Icon>
      <href>http://www.google.com/intl/en/images/logo.gif</href>
      <refreshMode>onInterval</refreshMode>
      <refreshInterval>86400</refreshInterval>
      <viewBoundScale>0.75</viewBoundScale>
   </Icon>
   <LatLonBox>
      <north>37.83234</north>
      <south>37.832122</south>
      <east>-122.373033</east>
      <west>-122.373724</west>
      <rotation>45</rotation>
   </LatLonBox>
</GroundOverlay>
</kml>
```

Extends

- *<Feature>*
- *<Overlay>*

Contained By

- <Document>
- <Folder>

<Icon>

Syntax

```
<Icon id="ID">
  <!-- specific to Icon -->
  <href>...</href>                        <!-- anyURI -->
  <gx:x>0<gx:x/>                          <!-- int -->
  <gx:y>0<gx:y/>                          <!-- int -->
  <gx:w>-1<gx:w/>                         <!-- int -->
  <gx:h>-1<gx:h/>                         <!-- int -->
  <refreshMode>onChange</refreshMode>
    <!-- kml:refreshModeEnum: onChange, onInterval, or onExpire -->
  <refreshInterval>4</refreshInterval>    <!-- float -->
  <viewRefreshMode>never</viewRefreshMode>
    <!-- kml:viewRefreshModeEnum: never, onStop, onRequest, onRegion -->
  <viewRefreshTime>4</viewRefreshTime>    <!-- float -->
  <viewBoundScale>1</viewBoundScale>      <!-- float -->
  <viewFormat>...</viewFormat>            <!-- string -->
  <httpQuery>...</httpQuery>              <!-- string -->
</Icon>
```

Description

Defines an image associated with an Icon style or overlay. The required <href> child element defines the location of the image to be used as the overlay or as the icon for the placemark. This location can either be on a local file system or a remote web server. The <gx:x>, <gx:y>, <gx:w>, and <gx:h> elements are used to select one icon from an image that contains multiple icons (often referred to as an *icon palette*.

```
<Icon>
```

```
  <href>Sunset.jpg</href>   <!-- Here, the image contains only one icon -->
</Icon>
```

```
<Icon>
  <href>/home/mydir/myiconpalette.jpg</href>
  <gx:w>138</gx:w>
  <gx:h>138</gx:h>
    <!-- Since x and y values are omitted, these measurements are made starting 
at
    the lower-left corner of the icon palette image -->
</Icon>
```

Elements Specific to Icon

<href>

An HTTP address or a local file specification used to load an icon.

<gx:x> and <gx:y>

If the <href> specifies an icon palette, these elements identify the offsets, in pixels, from the lower-left corner of the icon palette.If no values are specified for x and y, the lower left corner of the icon palette is assumed to be the lower-left corner of the icon to use.

<gx:w> and <gx:h>

If the <href> specifies an icon palette, these elements specify the width (<gx:w>) and height (<gx:h>), in pixels, of the icon to use.

<refreshMode>

For a description of <refreshMode> and the other elements listed below, see <Link>.

<refreshInterval>

<viewRefreshMode>

<viewRefreshTime>

<viewBoundScale>

<viewFormat>

<httpQuery>

Contained By

- <GroundOverlay>
- <ScreenOverlay>
- <IconStyle>

<IconStyle>

Syntax

```
<IconStyle id="ID">
  <!-- inherited from ColorStyle -->
  <color>ffffffff</color>            <!-- kml:color -->
  <colorMode>normal</colorMode>      <!-- kml:colorModeEnum:normal or random -->

  <!-- specific to IconStyle -->
  <scale>1</scale>                   <!-- float -->
  <heading>0</heading>               <!-- float -->
  <Icon>
    <href>...</href>
  </Icon>
  <hotSpot x="0.5"  y="0.5"
    xunits="fraction" yunits="fraction"/>    <!-- kml:vec2 -->
</IconStyle>
```

Description

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>.

Elements Specific to IconStyle

<scale>

Resizes the icon.

Note: The <geomScale> tag has been deprecated. Use <scale> instead.

<heading>

Direction (that is, North, South, East, West), in degrees. Default=0 (North). (See diagram.) Values range from 0 to 360 degrees.

<Icon>

A custom Icon. In <IconStyle>, the only child element of <Icon> is <href>:

- **<href>**: An HTTP address or a local file specification used to load an icon.

<hotSpot x="0.5" y="0.5" xunits="fraction" yunits="fraction">

Specifies the position within the Icon that is "anchored" to the <Point> specified in the Placemark. The *x* and *y* values can be specified in three different ways: as *pixels* (**"pixels"**), as *fractions* of the icon (**"fraction"***)*, or as *inset pixels* (**"insetPixels"**), which is an offset in pixels from the upper right corner of the icon. The *x* and *y* positions can be specified in different ways—for example, *x* can be in pixels and *y* can be a fraction. The origin of the coordinate system is in the lower left corner of the icon.

- **x** - Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the *x* component of a point on the icon.
- **y** - Either the number of pixels, a fractional component of the icon, or a pixel inset indicating the *y* component of a point on the icon.
- **xunits** - Units in which the *x* value is specified. A value of **fraction** indicates the *x* value is a fraction of the icon. A value of **pixels** indicates the *x* value in pixels. A value of **insetPixels** indicates the indent from the right edge of the icon.
- **yunits** - Units in which the *y* value is specified. A value of **fraction** indicates the *y* value is a fraction of the icon. A value of **pixels** indicates the *y* value in pixels. A value of **insetPixels** indicates the indent from the top edge of the icon.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
   <Style id="randomColorIcon">
      <IconStyle>
         <color>ff00ff00</color>
         <colorMode>random</colorMode>
         <scale>1.1</scale>
         <Icon>
            <href>http://maps.google.com/mapfiles/kml/pal3/icon21.png</href>
         </Icon>
      </IconStyle>
   </Style>
   <Placemark>
      <name>IconStyle.kml</name>
      <styleUrl>#randomColorIcon</styleUrl>
      <Point>
         <coordinates>-122.36868,37.831145,0</coordinates>
      </Point>
   </Placemark>
</Document>
</kml>
```

Extends

- *<ColorStyle>*

Contained By

- <Style>

Contains

- <href> as a child of <Icon>

<kml>

Syntax

```
<kml xmlns="http://www.opengis.net/kml/2.2" hint="target=sky"> ... </kml>
```

Description

The root element of a KML file. This element is required. It follows the xml declaration at the beginning of the file. The *hint* attribute is used as a signal to Google Earth to display the file as celestial data.

The <kml> element may also include the namespace for any external XML schemas that are referenced within the file.

A basic <kml> element contains 0 or 1 Feature and 0 or 1 NetworkLinkControl:

```
<kml xmlns="http://www.opengis.net/kml/2.2">
  <NetworkLinkControl> ... </NetworkLinkControl>
  <!-- 0 or 1 Feature elements -->
</kml>
```

<LabelStyle>

Syntax

```
<LabelStyle id="ID">
  <!-- inherited from ColorStyle -->
  <color>ffffffff</color>            <!-- kml:color -->
  <colorMode>normal</colorMode>      <!-- kml:colorModeEnum: normal or
random -->

  <!-- specific to LabelStyle -->
  <scale>1</scale>                   <!-- float -->
</LabelStyle>
```

Description

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.

Note: The <labelColor> tag is deprecated. Use <LabelStyle> instead.

Specific to <LabelStyle>

<scale>

Resizes the label.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
   <Style id="randomLabelColor">
      <LabelStyle>
         <color>ff0000cc</color>
         <colorMode>random</colorMode>
         <scale>1.5</scale>
      </LabelStyle>
   </Style>
   <Placemark>
      <name>LabelStyle.kml</name>
      <styleUrl>#randomLabelColor</styleUrl>
      <Point>
         <coordinates>-122.367375,37.829192,0</coordinates>
      </Point>
   </Placemark>
</Document>
</kml>
```

Extends

- *<ColorStyle>*

Contained By

- <Style>

<gx:LatLonQuad>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<GroundOverlay id="ID">
  ...
  <Icon>...</Icon>
  <altitude>0</altitude>
  <altitudeMode>clampToGround</altitudeMode>                   <!-- or
absolute -->
       <!-- can substitute
<gx:altitudeMode>clampToSeaFloor</gx:altitudeMode> -->

  <gx:LatLonQuad>
    <coordinates>...</coordinates>              <!-- four lon,lat tuples -->
  </gx:LatLonQuad>
</GroundOverlay>
```

Description

Allows nonrectangular quadrilateral ground overlays.

Specifies the coordinates of the four corner points of a quadrilateral defining the overlay area. Exactly four coordinate tuples have to be provided, each consisting of floating point values for longitude and latitude. Insert a space between tuples. Do not include spaces within a tuple. The coordinates must be specified in counter-clockwise order with the first coordinate corresponding to the lower-left corner of the overlayed image. The shape described by these corners must be convex.

If a third value is inserted into any tuple (representing altitude) it will be ignored. Altitude is set using `<altitude>` and `<altitudeMode>` (or `<gx:altitudeMode>`) extending `<GroundOverlay>`. Allowed altitude modes are **absolute**, **clampToGround**, and **clampToSeaFloor**.

Example

```
<?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">
<GroundOverlay>
  <name>gx:LatLonQuad Example</name>
  <Icon>
```

```
<href>http://code.google.com/apis/kml/documentation/Images/rectangle.gif</href>
    <viewBoundScale>0.75</viewBoundScale>
  </Icon>
  <gx:LatLonQuad>
    <coordinates>
      81.601884,44.160723 83.529902,43.665148 82.947737,44.248831
81.509322,44.321015
    </coordinates>
  </gx:LatLonQuad>
</GroundOverlay>
</kml>
```

Extends

- <Object>

Contained by

- <GroundOverlay>

<LinearRing>

Syntax

```
<LinearRing id="ID">
  <!-- specific to LinearRing -->
  <gx:altitudeOffset>0</gx:altitudeOffset>   <!-- double -->
  <extrude>0</extrude>                       <!-- boolean -->
  <tessellate>0</tessellate>                 <!-- boolean -->
  <altitudeMode>clampToGround</altitudeMode>
    <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or absolute -->
    <!-- or, substitute gx:altitudeMode: clampToSeaFloor, relativeToSeaFloor --
>
  <coordinates>...</coordinates>             <!-- lon,lat[,alt] tuples -->
</LinearRing>
```

Description

Defines a closed line string, typically the outer boundary of a Polygon. Optionally, a LinearRing can also be used as the inner boundary of a Polygon to create holes in the Polygon. A Polygon can contain multiple <LinearRing> elements used as inner boundaries.

Note: In Google Earth, a Polygon with an <altitudeMode> of *clampToGround* follows lines of constant bearing; however, a LinearRing (by itself) with an <altitudeMode> of *clampToGround* follows great circle lines.

Elements Specific to LinearRing

<gx:altitudeOffset>

A KML extension, in the Google extension namespace, that modifies how the altitude values are rendered. This offset allows you to move an entire LinearRing up or down as a unit without modifying all the individual coordinate values that make up the LinearRing. (Although the LinearRing is displayed using the altitude offset value, the original altitude values are preserved in the KML file.) Units are in meters.

<extrude>

Boolean value. Specifies whether to connect the LinearRing to the ground. To extrude this geometry, the altitude mode must be either **relativeToGround**, **relativeToSeaFloor**, or **absolute**. Only the vertices of the LinearRing are extruded, not the center of the geometry. The vertices are extruded toward the center of the Earth's sphere.

<tessellate>

Boolean value. Specifies whether to allow the LinearRing to follow the terrain. To enable tessellation, the value for <altitudeMode> must be **clampToGround** or **clampToSeaFloor**. Very large LinearRings should enable tessellation so that they follow the curvature of the earth (otherwise, they may go underground and be hidden).

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround -** (default) Indicates to ignore an altitude specification (for example, in the <coordinates> tag).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters. A typical use of this mode is for placing telephone poles or a ski lift.
- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an *absolute* altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters. A typical use of this mode is for aircraft placement.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<coordinates> ***(required)***

Four or more tuples, each consisting of floating point values for *longitude*, *latitude*, and *altitude*. The *altitude* component is optional. Do not include spaces within a tuple. The last coordinate must be the same as the first coordinate. Coordinates are expressed in decimal degrees only.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Placemark>
  <name>LinearRing.kml</name>
  <Polygon>
    <outerBoundaryIs>
      <LinearRing>
        <coordinates>
          -122.365662,37.826988,0
          -122.365202,37.826302,0
          -122.364581,37.82655,0
          -122.365038,37.827237,0
          -122.365662,37.826988,0
        </coordinates>
      </LinearRing>
    </outerBoundaryIs>
  </Polygon>
</Placemark>
</kml>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <Placemark>
- <innerBoundaryIs>
- <outerBoundaryIs>

<LineString>

Syntax

```
<LineString id="ID">
```

```
  <!-- specific to LineString -->
  <gx:altitudeOffset>0</gx:altitudeOffset>  <!-- double -->
  <extrude>0</extrude>                      <!-- boolean -->
  <tessellate>0</tessellate>                <!-- boolean -->
  <altitudeMode>clampToGround</altitudeMode>
      <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or absolute -
->
      <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <gx:drawOrder>0</gx:drawOrder>            <!-- integer -->
  <coordinates>...</coordinates>            <!-- lon,lat[,alt] -->
</LineString>
```

Description

Defines a connected set of line segments. Use <LineStyle> to specify the color, color mode, and width of the line. When a LineString is extruded, the line is extended to the ground, forming a polygon that looks somewhat like a wall or fence. For extruded LineStrings, the line itself uses the current LineStyle, and the extrusion uses the current PolyStyle. See the *KML Tutorial* for examples of LineStrings (or *paths*).

Elements Specific to LineString

<gx:altitudeOffset>

A KML extension, in the Google extension namespace, that modifies how the altitude values are rendered. This offset allows you to move an entire LineString up or down as a unit without modifying all the individual coordinate values that make up the LineString. (Although the LineString is displayed using the altitude offset value, the original altitude values are preserved in the KML file.) Units are in meters.

<extrude>

Boolean value. Specifies whether to connect the LineString to the ground. To extrude a LineString, the altitude mode must be either **relativeToGround**, **relativeToSeaFloor**, or **absolute**. The vertices in the LineString are extruded toward the center of the Earth's sphere.

<tessellate>

Boolean value. Specifies whether to allow the LineString to follow the terrain. To enable tessellation, the altitude mode must be **clampToGround** or **clampToSeaFloor**. Very large LineStrings should enable tessellation so that they follow the curvature of the earth (otherwise, they may go underground and be hidden).

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore an altitude specification (for example, in the <coordinates> tag).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters. A typical use of this mode is for placing telephone poles or a ski lift.
- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an *absolute* altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters. A typical use of this mode is for aircraft placement.

<gx:altitudeMode>

A KML extension, in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<gx:drawOrder>

An integer value that specifies the order for drawing multiple line strings. LineStrings drawn first may be partially or fully obscured by LineStrings with a later draw order. This element may be required in conjunction with the `<gx:outerColor>` and `<gx:outerWidth>` elements in <LineStyle> when dual-colored lines cross each other.

<coordinates> ***(required)***

Two or more coordinate tuples, each consisting of floating point values for *longitude*, *latitude*, and *altitude*. The *altitude* component is optional. Insert a space between tuples. Do not include spaces within a tuple.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>LineString.kml</name>
  <open>1</open>
  <LookAt>
    <longitude>-122.36415</longitude>
    <latitude>37.824553</latitude>
    <altitude>0</altitude>
    <range>150</range>
    <tilt>50</tilt>
    <heading>0</heading>
  </LookAt>
  <Placemark>
    <name>unextruded</name>
    <LineString>
      <extrude>1</extrude>
      <tessellate>1</tessellate>
      <coordinates>
        -122.364383,37.824664,0 -122.364152,37.824322,0
      </coordinates>
    </LineString>
  </Placemark>
  <Placemark>
    <name>extruded</name>
    <LineString>
      <extrude>1</extrude>
      <tessellate>1</tessellate>
      <altitudeMode>relativeToGround</altitudeMode>
      <coordinates>
        -122.364167,37.824787,50 -122.363917,37.824423,50
      </coordinates>
    </LineString>
  </Placemark>
</Document>
</kml>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <Placemark>

<LineStyle>

Syntax

```
<LineStyle id="ID">
  <!-- inherited from ColorStyle -->
  <color>ffffffff</color>            <!-- kml:color -->
  <colorMode>normal</colorMode>      <!-- colorModeEnum: normal or
random -->

  <!-- specific to LineStyle -->
  <width>1</width>                   <!-- float -->
  <gx:outerColor>ffffffff</gx:outerColor>     <!-- kml:color -->
  <gx:outerWidth>0.0</gx:outerWidth>          <!-- float -->
  <gx:physicalWidth>0.0</gx:physicalWidth>    <!-- float -->
</LineStyle>
```

Description

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).

Elements Specific to LineStyle

<width>

Width of the line, in pixels.

<gx:outerColor>

Color of the portion of the line defined by `<gx:outerWidth>`. Note that the <gx:outerColor> and <gx:outerWidth> elements are ignored when <LineStyle> is applied to <Polygon> and <LinearRing>.

<gx:outerWidth>

A value between 0.0 and 1.0 that specifies the proportion of the line that uses the `<gx:outerColor>`. See also `<gx:drawOrder>` in `<LineString>`. A draw order value may be necessary if dual-colored lines are crossing each other—for example, for showing freeway interchanges.

<gx:physicalWidth>

Physical width of the line, in meters.

Example

The following example shows a 50 percent opaque red line with a width of 4 pixels.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>LineStyle.kml</name>
  <open>1</open>
  <Style id="linestyleExample">
    <LineStyle>
      <color>7f0000ff</color>
      <width>4</width>
    </LineStyle>
  </Style>
  <Placemark>
    <name>LineStyle Example</name>
    <styleUrl>#linestyleExample</styleUrl>
    <LineString>
      <extrude>1</extrude>
      <tessellate>1</tessellate>
      <coordinates>
        -122.364383,37.824664,0 -122.364152,37.824322,0
      </coordinates>
    </LineString>
  </Placemark>
</Document>
</kml>
```

Extends

- *<ColorStyle>*

Contained By

- <Style>

<Link>

Syntax

```
<Link id="ID">
  <!-- specific to Link -->
  <href>...</href>                      <!-- string -->
  <refreshMode>onChange</refreshMode>
    <!-- refreshModeEnum: onChange, onInterval, or onExpire -->
  <refreshInterval>4</refreshInterval>  <!-- float -->
  <viewRefreshMode>never</viewRefreshMode>
    <!-- viewRefreshModeEnum: never, onStop, onRequest, onRegion -->
  <viewRefreshTime>4</viewRefreshTime>  <!-- float -->
  <viewBoundScale>1</viewBoundScale>    <!-- float -->
  <viewFormat>BBOX=[bboxWest],[bboxSouth],[bboxEast],[bboxNorth]</viewFormat>
                                        <!-- string -->
  <httpQuery>...</httpQuery>            <!-- string -->
</Link>
```

Description

<Link> specifies the location of any of the following:

- KML files fetched by network links
- Image files used in any Overlay (the <Icon> element specifies the image in an Overlay; <Icon> has the same fields as <Link>)
- Model files used in the <Model> element

The file is conditionally loaded and refreshed, depending on the refresh parameters supplied here. Two different sets of refresh parameters can be specified: one set is based on *time* (<refreshMode> and <refreshInterval>) and one is based on the current "camera" *view* (<viewRefreshMode> and <viewRefreshTime>). In addition, Link specifies whether to scale the bounding box parameters that are sent to the server (<viewBoundScale> and provides a set of optional viewing parameters that can be sent to the server (<viewFormat>) as well as a set of optional parameters containing version and language information.

When a file is fetched, the URL that is sent to the server is composed of three pieces of information:

- the *href* (Hypertext Reference) that specifies the file to load.
- an *arbitrary format string* that is created from (a) parameters that you specify in the <viewFormat> element or (b) bounding box parameters (this is the default and is used if no <viewFormat> element is included in the file).
- a second *format string that is specified in the <httpQuery>* element.

If the file specified in <href> is a local file, the <viewFormat> and <httpQuery> elements are not used.

The <Link> element replaces the <Url> element of <NetworkLink> contained in earlier KML releases and adds functionality for the <Region> element (introduced in KML 2.1). In Google Earth releases 3.0 and earlier, the <Link> element is ignored.

Elements Specific to Link

<href>

A URL (either an HTTP address or a local file specification). When the parent of <Link> is a NetworkLink, <href> is a KML file. When the parent of <Link> is a Model, <href> is a COLLADA file. When the parent of <Icon> (same fields as <Link>) is an Overlay, <href> is an image. Relative URLs can be used in this tag and are evaluated relative to the enclosing KML file. See KMZ Files for details on constructing relative references in KML and KMZ files.

<refreshMode>

Specifies a time-based refresh mode, which can be one of the following:

- **onChange** - refresh when the file is loaded and whenever the Link parameters change (the default).
- **onInterval** - refresh every *n* seconds (specified in *<refreshInterval>*).
- **onExpire** - refresh the file when the expiration time is reached. If a fetched file has a NetworkLinkControl, the <expires> time takes precedence over expiration times specified in HTTP headers. If no *<expires>* time is specified, the HTTP *max-age* header is used (if present). If *max-age* is not present, the *Expires* HTTP header is used (if present). (See Section RFC261b of the *Hypertext Transfer Protocol - HTTP 1.1* for details on HTTP header fields.)

<refreshInterval>

Indicates to refresh the file every *n* seconds.

<viewRefreshMode>

Specifies how the link is refreshed when the "camera" changes.

Can be one of the following:

- **never** (default) - Ignore changes in the view. Also ignore <viewFormat> parameters, if any.
- **onStop** - Refresh the file *n* seconds after movement stops, where *n* is specified in <viewRefreshTime>.
- **onRequest** - Refresh the file only when the user explicitly requests it. (For example, in Google Earth, the user right-clicks and selects Refresh in the Context menu.)
- **onRegion** - Refresh the file when the Region becomes active. See <Region>.

<viewRefreshTime>

After camera movement stops, specifies the number of seconds to wait before refreshing the view. (See <viewRefreshMode> and **onStop** above.)

<viewBoundScale>

Scales the BBOX parameters before sending them to the server. A value less than 1 specifies to use less than the full view (screen). A value greater than 1 specifies to fetch an area that extends beyond the edges of the current view.

<viewFormat>

Specifies the format of the query string that is appended to the Link's <href> before the file is fetched.(If the <href> specifies a local file, this element is ignored.)

If you specify a <viewRefreshMode> of **onStop** and do not include the <viewFormat> tag in the file, the following information is automatically appended to the query string:

BBOX=[bboxWest],[bboxSouth],[bboxEast],[bboxNorth]

This information matches the Web Map Service (WMS) bounding box specification.

If you specify an empty <viewFormat> tag, no information is appended to the query string.

You can also specify a custom set of viewing parameters to add to the query string. If you supply a format string, it is used ***instead of*** the BBOX information. If you also want the BBOX information, you need to add those parameters along with the custom parameters.

You can use any of the following parameters in your format string (and Google Earth will substitute the appropriate current value at the time it creates the query string):

- **[lookatLon]**, **[lookatLat]** - longitude and latitude of the point that <LookAt> is viewing
- **[lookatRange]**, **[lookatTilt]**, **[lookatHeading]** - values used by the <LookAt> element (see descriptions of <range>, <tilt>, and <heading> in <LookAt>)
- **[lookatTerrainLon], [lookatTerrainLat], [lookatTerrainAlt]** - point on the terrain in degrees/meters that <LookAt> is viewing
- **[cameraLon], [cameraLat], [cameraAlt]** - degrees/meters of the eyepoint for the camera
- **[horizFov]**, **[vertFov]** - horizontal, vertical field of view for the camera
- **[horizPixels]**, **[vertPixels]** - size in pixels of the 3D viewer
- **[terrainEnabled]** - indicates whether the 3D viewer is showing terrain

<httpQuery>

Appends information to the query string, based on the parameters specified. (Google Earth substitutes the appropriate current value at the time it creates the query string.) The following parameters are supported:

- **[clientVersion]**
- **[kmlVersion]**
- **[clientName]**
- **[language]**

Example

```
<NetworkLink>
  <name>NE US Radar</name>
  <flyToView>1</flyToView>
  <Link>

<href>http://www.example.com/geotiff/NE/MergedReflectivityQComposite.kml</href
>
    <refreshMode>onInterval</refreshMode>
    <refreshInterval>30</refreshInterval>
    <viewRefreshMode>onStop</viewRefreshMode>
    <viewRefreshTime>7</viewRefreshTime>
    <viewFormat>BBOX=[bboxWest],[bboxSouth],[bboxEast],[bboxNorth];CAMERA=\
      [lookatLon],[lookatLat],[lookatRange],[lookatTilt],[lookatHeading];VIEW=\
      [horizFov],[vertFov],[horizPixels],[vertPixels],[terrainEnabled]
</viewFormat>
  </Link>
</NetworkLink>
```

Extends

- <Object>

Contained By

- <Model>
- <NetworkLink>

See Also

- <NetworkLinkControl>
- <Region>

<ListStyle>

Syntax

```
<ListStyle id="ID">
  <!-- specific to ListStyle -->
  <listItemType>check</listItemType> <!-- kml:listItemTypeEnum:check,
                                          checkOffOnly,checkHideChildren,
                                         radioFolder -->
  <bgColor>ffffffff</bgColor>        <!-- kml:color -->
  <ItemIcon>                         <!-- 0 or more ItemIcon elements -->
    <state>open</state>
      <!-- kml:itemIconModeEnum:open, closed, error, fetching0, fetching1, or
fetching2 -->
    <href>...</href>                 <!-- anyURI -->
  </ItemIcon>
</ListStyle>
```

Description

Specifies how a Feature is displayed in the list view. The *list view* is a hierarchy of containers and children; in Google Earth, this is the Places panel.

Elements Specific to ListStyle

<listItemType>

Specifies how a Feature is displayed in the list view. Possible values are:

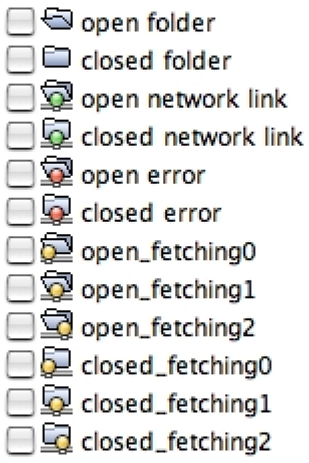
- **check** (default) - The Feature's visibility is tied to its item's checkbox.
- **radioFolder** - When specified for a Container, only one of the Container's items is visible at a time
- **checkOffOnly** - When specified for a Container or Network Link, prevents all items from being made visible at once—that is, the user can turn everything in the Container or Network Link off but cannot turn everything on at the same time. This setting is useful for Containers or Network Links containing large amounts of data.
- **checkHideChildren** - Use a normal checkbox for visibility but do not display the Container or Network Link's children in the list view. A checkbox allows the user to toggle visibility of the child objects in the viewer.

<bgColor>

Background color for the Snippet. Color and opacity values are expressed in hexadecimal notation. The range of values for any one color is 0 to 255 (`00` to `ff`). For alpha, `00` is fully transparent and `ff` is fully opaque. The order of expression is *aabbggrr,* where *aa=alpha* (00 to ff); *bb=blue* (00 to ff); *gg=green* (00 to ff); *rr=red* (00 to ff). For example, if you want to apply a blue color with 50 percent opacity to an overlay, you would specify the following: `<color>7fff0000</color>`, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00.

<ItemIcon>

Icon used in the List view that reflects the state of a Folder or Link fetch. Icons associated with the **open** and **closed** modes are used for Folders and Network Links. Icons associated with the **error** and **fetching0**, **fetching1**, and **fetching2** modes are used for Network Links. The following screen capture illustrates the Google Earth icons for these states:



<state>

Specifies the current state of the NetworkLink or Folder. Possible values are **open**, **closed**, **error**, **fetching0**, **fetching1**, and **fetching2**. These values can be combined by inserting a space between two values (no comma).

<href>

Specifies the URI of the image used in the List View for the Feature.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>ListStyle.kml</name>
  <open>1</open>
  <Style id="bgColorExample">
    <ListStyle>
      <bgColor>ff336699</bgColor>
    </ListStyle>
  </Style>
  <Style id="checkHideChildrenExample">
    <ListStyle>
      <listItemType>checkHideChildren</listItemType>
    </ListStyle>
  </Style>
  <Style id="radioFolderExample">
    <ListStyle>
      <listItemType>radioFolder</listItemType>
    </ListStyle>
  </Style>
  <Folder>
    <name>ListStyle Examples</name>
    <open>1</open>
    <Folder>
      <name>bgColor example</name>
      <open>1</open>
      <Placemark>
        <name>pl1</name>
        <Point>
          <coordinates>-122.362815,37.822931,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl2</name>
        <Point>
          <coordinates>-122.362825,37.822931,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl3</name>
        <Point>
          <coordinates>-122.362835,37.822931,0</coordinates>
        </Point>
      </Placemark>
      <styleUrl>#bgColorExample</styleUrl>
    </Folder>
    <Folder>
      <name>checkHideChildren example</name>
      <open>1</open>
      <Placemark>
        <name>pl4</name>
        <Point>
          <coordinates>-122.362845,37.822941,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl5</name>
        <Point>
          <coordinates>-122.362855,37.822941,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl6</name>
        <Point>
          <coordinates>-122.362865,37.822941,0</coordinates>
        </Point>
      </Placemark>
      <styleUrl>#checkHideChildrenExample</styleUrl>
    </Folder>
    <Folder>
      <name>radioFolder example</name>
      <open>1</open>
      <Placemark>
        <name>pl7</name>
        <Point>
          <coordinates>-122.362875,37.822951,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl8</name>
        <Point>
          <coordinates>-122.362885,37.822951,0</coordinates>
        </Point>
      </Placemark>
      <Placemark>
        <name>pl9</name>
        <Point>
          <coordinates>-122.362895,37.822951,0</coordinates>
        </Point>
      </Placemark>
      <styleUrl>#radioFolderExample</styleUrl>
    </Folder>
  </Folder>
</Document>
</kml>
```

Extends

- <Object>

Contained By

- <Style>

<LookAt>

Syntax

```
<LookAt id="ID">
  <!-- inherited from AbstractView element -->
  <TimePrimitive>...</TimePrimitive>  <!-- gx:TimeSpan or gx:TimeStamp -->
    <gx:ViewerOptions>
    <option> name=" " type="boolean">     <!--
name="streetview", "historicalimagery", "sunlight", or "groundnavigation" -->
    </option>
  </gx:ViewerOptions>

  <!-- specific to LookAt -->
  <longitude>0</longitude>            <!-- kml:angle180 -->
  <latitude>0</latitude>              <!-- kml:angle90 -->
  <altitude>0</altitude>              <!-- double -->
  <heading>0</heading>                <!-- kml:angle360 -->
  <tilt>0</tilt>                      <!-- kml:anglepos90 -->
  <range></range>                     <!-- double -->
  <altitudeMode>clampToGround</altitudeMode>
          <!--kml:altitudeModeEnum:clampToGround, relativeToGround, absolute -
->
          <!-- or, gx:altitudeMode can be substituted: clampToSeaFloor,
relativeToSeaFloor -->

</LookAt>
```

Description

Defines a virtual camera that is associated with any element derived from Feature. The LookAt element positions the "camera" in relation to the object that is being viewed. In Google Earth, the view "flies to" this LookAt viewpoint when the user double-clicks an item in the Places panel or double-clicks an icon in the 3D viewer.

Elements Specific to LookAt

<longitude>

Longitude of the point the camera is looking at. Angular distance in degrees, relative to the Prime Meridian. Values west of the Meridian range from −180 to 0 degrees. Values east of the Meridian range from 0 to 180 degrees.

<latitude>

Latitude of the point the camera is looking at. Degrees north or south of the Equator (0 degrees). Values range from −90 degrees to 90 degrees.

<altitude>

Distance from the earth's surface, in meters. Interpreted according to the LookAt's altitude mode.

<heading>

Direction (that is, North, South, East, West), in degrees. Default=0 (North). (See diagram below.) Values range from 0 to 360 degrees.

<tilt>

Angle between the direction of the LookAt position and the normal to the surface of the earth. (See diagram below.) Values range from 0 to 90 degrees. Values for <tilt> cannot be negative. A <tilt> value of 0 degrees indicates viewing from directly above. A <tilt> value of 90 degrees indicates viewing along the horizon.

<range> ***(required)***

Distance in meters from the point specified by <longitude>, <latitude>, and <altitude> to the LookAt position. (See diagram below.)

<altitudeMode>

Specifies how the <altitude> specified for the LookAt point is interpreted. Possible values are as follows:

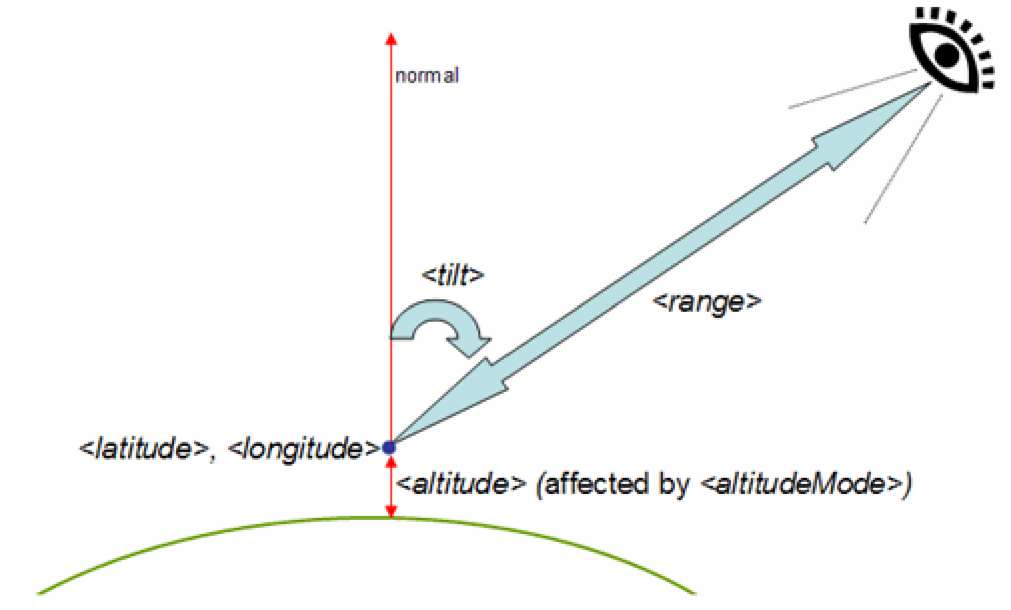
- **clampToGround** - (default) Indicates to ignore the <altitude> specification and place the LookAt position on the ground.
- **relativeToGround** - Interprets the <altitude> as a value in meters above the ground.
- **absolute** - Interprets the <altitude> as a value in meters above sea level.

<gx:altitudeMode>

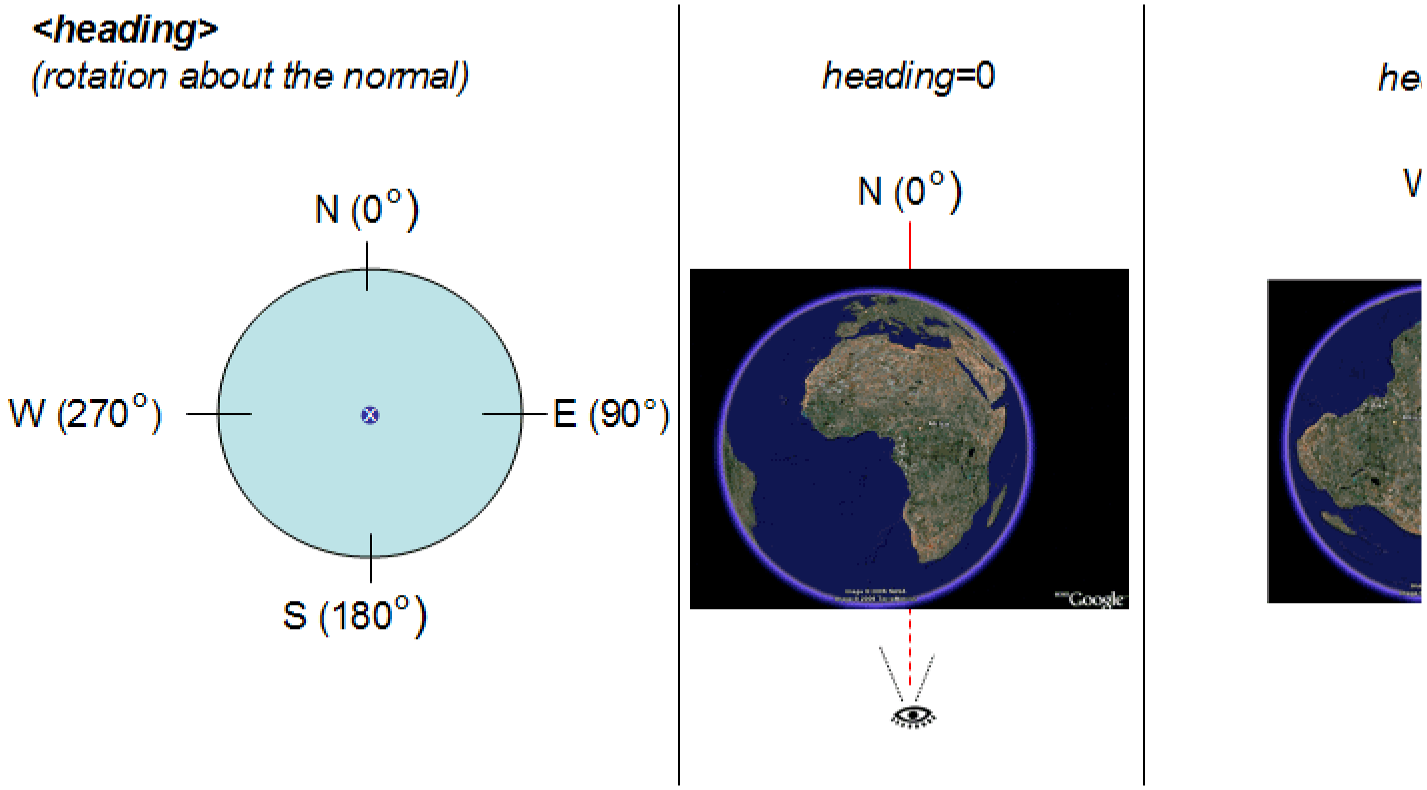
A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the LookAt will be positioned on the sea floor. If the point is on land rather than at sea, the LookAt will be positioned on the ground.

This diagram illustrates the <range>, <tilt>, and <altitude> elements:



This diagram illustrates the <heading> element:



```
<?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">

  <Placemark>
    <name>LookAt.kml</name>
    <LookAt>
      <gx:TimeStamp>
        <when>1994</when>
      </gx:TimeStamp>
      <longitude>-122.363</longitude>
      <latitude>37.81</latitude>
      <altitude>2000</altitude>
      <range>500</range>
      <tilt>45</tilt>
      <heading>0</heading>
      <altitudeMode>relativeToGround</altitudeMode>
    </LookAt>
    <Point>
      <coordinates>-122.363,37.82,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Extends

- *<AbstractView>*

Contained By

- Any element derived from *<Feature>*
- <NetworkLinkControl>

<Model>

Syntax

```
<Model id="ID">
  <!-- specific to Model -->
  <altitudeMode>clampToGround</altitudeMode>
      <!-- kml:altitudeModeEnum: clampToGround,relativeToGround,or absolute --
>
      <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <Location>
    <longitude></longitude> <!-- kml:angle180 -->
    <latitude></latitude>   <!-- kml:angle90 -->
    <altitude>0</altitude>  <!-- double -->
  </Location>
  <Orientation>
    <heading>0</heading>    <!-- kml:angle360 -->
    <tilt>0</tilt>          <!-- kml:angle360 -->
    <roll>0</roll>          <!-- kml:angle360 -->
  </Orientation>
  <Scale>
    <x>1</x>                <!-- double -->
    <y>1</y>                <!-- double -->
    <z>1</z>                <!-- double -->
  </Scale>
  <Link>...</Link>
  <ResourceMap>
    <Alias>
      <targetHref>...</targetHref>   <!-- anyURI -->
      <sourceHref>...</sourceHref>   <!-- anyURI -->
    </Alias>
  </ResourceMap>
</Model>
```

Description

A 3D object described in a COLLADA file (referenced in the <Link> tag). COLLADA files have a *.dae* file extension. Models are created in their own coordinate space and then located, positioned, and scaled in Google Earth. See the "Topics in KML" page on Regions for more detail.

Google Earth supports the COLLADA common profile, with the following exceptions:

- Google Earth supports only triangles and lines as primitive types. The maximum number of triangles allowed is 21845.
- Google Earth does not support animation or skinning.
- Google Earth does not support external geometry references.

Elements Specific to Model

<altitudeMode>

Specifies how the <altitude> specified in <Location> is interpreted. Possible values are as follows:

- **clampToGround** - (default) Indicates to ignore the <altitude> specification and place the Model on the ground.
- **relativeToGround** - Interprets the <altitude> as a value in meters above the ground.
- **absolute** - Interprets the <altitude> as a value in meters above sea level.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
- **clampToSeaFloor** - The <altitude> specification is ignored, and the Model will be positioned on the sea floor. If the point is on land rather than at sea, the Model will be positioned on the ground.

<Location>

Specifies the exact coordinates of the Model's origin in latitude, longitude, and altitude. **Latitude** and **longitude** measurements are standard lat-lon projection with WGS84 datum. **Altitude** is distance above the earth's surface, in meters, and is interpreted according to <altitudeMode> or <gx:altitudeMode>.

```
<Location>
  <longitude>39.55375305703105</longitude>
  <latitude>-118.9813220168456</latitude>
  <altitude>1223</altitude>
</Location>
```

<Orientation>

Describes rotation of a 3D model's coordinate system to position the object in Google Earth. See diagrams below.

```
<Orientation>
  <heading>45.0</heading>
  <tilt>10.0</tilt>
  <roll>0.0</roll>
</Orientation>
```

Rotations are applied to a Model in the following order:

1. <roll>
2. <tilt>
3. <heading>

<heading>

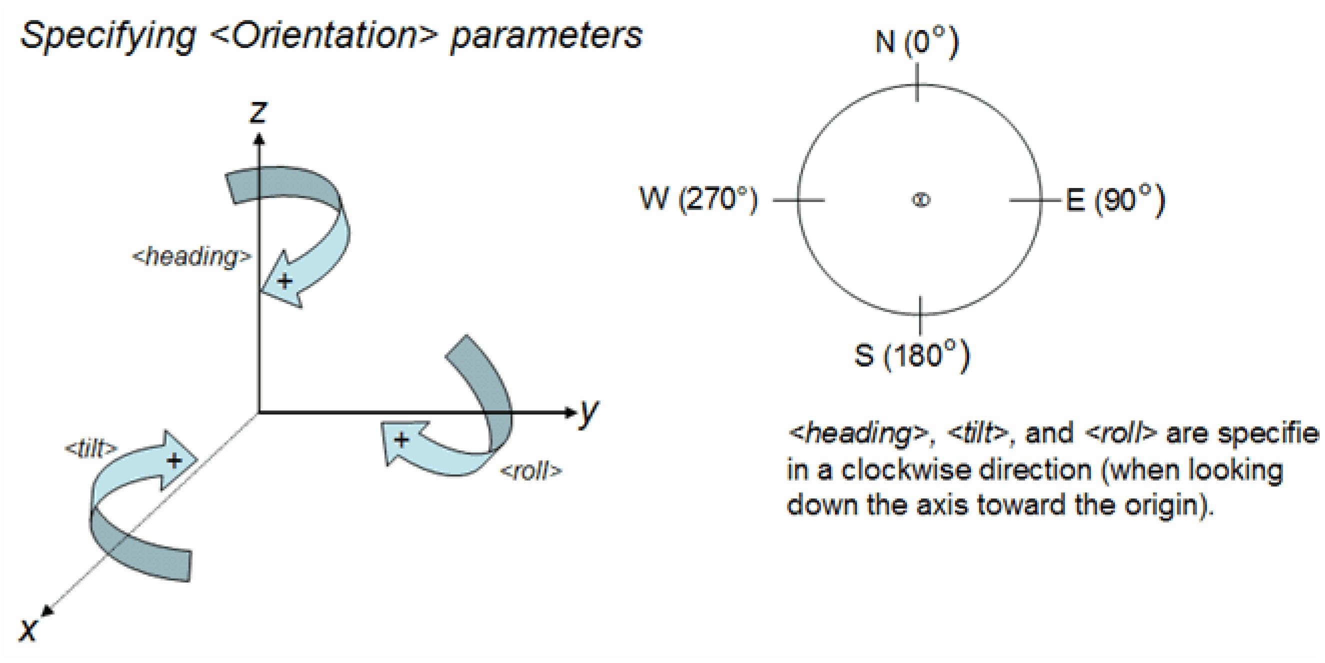
Rotation about the *z* axis (normal to the Earth's surface). A value of 0 (the default) equals North. A positive rotation is clockwise around the *z* axis and specified in degrees from 0 to 360.

<tilt>

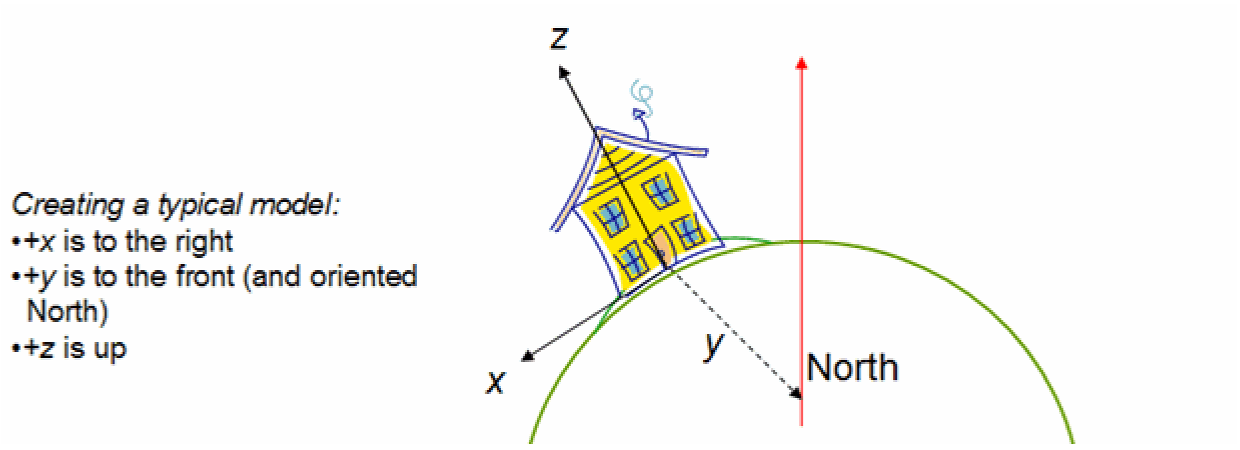
Rotation about the *x* axis. A positive rotation is clockwise around the *x* axis and specified in degrees from 0 to 360.

<roll>

Rotation about the *y* axis. A positive rotation is clockwise around the y axis and specified in degrees from 0 to 360.


Specifying <Orientation> parameters
z
<heading>
+
y
<roll>
+
<tilt>
+
x
N (0°)
W (270°)
E (90°)
S (180°)
<heading>, <tilt>, and <roll> are specifie
in a clockwise direction (when looking
down the axis toward the origin).

This diagram illustrates the typical orientation of a model's axes:


Creating a typical model:
•+x is to the right
•+y is to the front (and oriented North)
•+z is up
z
x
y
North

<Scale>

Scales a model along the *x*, *y*, and *z* axes in the model's coordinate space.

```
<Scale>
  <x>2.5</x>
  <y>2.5</y>
  <z>3.5</z>
</Scale>
```

<Link>

Specifies the file to load and optional refresh parameters. See <Link>.

<ResourceMap>

Specifies 0 or more **<Alias>** elements, each of which is a mapping for the texture file path from the original Collada file to the KML or KMZ file that contains the Model. This element allows you to move and rename texture files without having to update the original Collada file that references those textures. One <ResourceMap> element can contain multiple mappings from different (source) Collada files into the same (target) KMZ file.

```
<Alias>
  <targetHref>../images/foo.jpg</targetHref>
  <sourceHref>c:\mytextures\foo.jpg</sourceHref>
</Alias>
```

<Alias> contains a mapping from a <sourceHref> to a <targetHref>:

<targetHref>

Specifies the texture file to be fetched by Google Earth. This reference can be a relative reference to an image file within the *.kmz* archive, or it can be an absolute reference to the file (for example, a URL).

<sourceHref>

Is the path specified for the texture file in the Collada *.dae* file.

In Google Earth, if this mapping is not supplied, the following rules are used to locate the textures referenced in the Collada (*.dae*) file:

- ***No path:*** If the texture name does not include a path, Google Earth looks for the texture in the same directory as the *.dae* file that references it.
- ***Relative path:*** If the texture name includes a relative path (for example, *../images/mytexture.jpg*), Google Earth interprets the path as being relative to the *.dae* file that references it.
- ***Absolute path:*** If the texture name is an absolute path (*c:\mytexture.jpg*) or a network path (for example, *http://myserver.com/mytexture.jpg*), Google Earth looks for the file in the specified location, regardless of where the *.dae* file is located.

Example

```
<Model id="khModel543">
```

```
  <altitudeMode>relativeToGround</altitudeMode>
  <Location>
    <longitude>39.55375305703105</longitude>
    <latitude>-118.9813220168456</latitude>
    <altitude>1223</altitude>
  </Location>
  <Orientation>
    <heading>45.0</heading>
    <tilt>10.0</tilt>
    <roll>0.0</roll>
  </Orientation>
  <Scale>
    <x>1.0</x>
    <y>1.0</y>
    <z>1.0</z>
  </Scale>
  <Link>
    <href>house.dae</href>
    <refreshMode>once</refreshMode>
  </Link>
  <ResourceMap>
    <Alias>
      <targetHref>../files/CU-Macky---Center-
StairsnoCulling.jpg</targetHref>
      <sourceHref>CU-Macky---Center-
StairsnoCulling.jpg</sourceHref>
    </Alias>
    <Alias>
      <targetHref>../files/CU-Macky-
4sideturretnoCulling.jpg</targetHref>
      <sourceHref>CU-Macky-4sideturretnoCulling.jpg</sourceHref>
    </Alias>
    <Alias>
      <targetHref>../files/CU-Macky-Back-
NorthnoCulling.jpg</targetHref>
      <sourceHref>CU-Macky-Back-NorthnoCulling.jpg</sourceHref>
    </Alias>
  </ResourceMap>
</Model>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <Placemark>

<MultiGeometry>

Syntax

```
<MultiGeometry id="ID">
  <!-- specific to MultiGeometry -->
  <!-- 0 or more Geometry elements -->
</MultiGeometry>
```

Description

A container for zero or more geometry primitives associated with the same feature.

Note: The <GeometryCollection> tag has been deprecated. Use <MultiGeometry> instead.

Elements Specific to MultiGeometry

- 0 or more <Geometry> elements

Example

```
<Placemark>
  <name>SF Marina Harbor Master</name>
  <visibility>0</visibility>
  <MultiGeometry>
    <LineString>
      <!-- north wall -->
      <coordinates>
        -122.4425587930444,37.80666418607323,0
        -122.4428379594768,37.80663578323093,0
      </coordinates>
    </LineString>
    <LineString>
      <!-- south wall -->
      <coordinates>
        -122.4425509770566,37.80662588061205,0
        -122.4428340530617,37.8065999493009,0
      </coordinates>
    </LineString>
  </MultiGeometry>
</Placemark>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <Placemark>

<gx:MultiTrack>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.2 and later. Learn more

Syntax

```
<gx:MultiTrack id="ID">
  <!-- specific to MultiTrack -->
  <altitudeMode>clampToGround</altitudeMode>
        <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or
absolute -->
        <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <gx:interpolate>0<gx:interpolate>   <!-- boolean -->
  <gx:Track>...</gx:Track>            <!-- one or more gx:Track elements -->
</gx:MultiTrack>
```

Description

A multi-track element is used to combine multiple track elements into a single conceptual unit. For example, suppose you collect GPS data for a day's bike ride that includes several rest stops and a stop for lunch. Because of the interruptions in time, one bike ride might appear as four different tracks when the times and positions are plotted. Grouping these <gx:Track> elements into one <gx:MultiTrack> container causes them to be displayed in Google Earth as sections of a single path. When the icon reaches the end of one segment, it moves to the beginning of the next segment. The <gx:interpolate> element specifies whether to stop at the end of one track and jump immediately to the start of the next one, or to interpolate the missing values between the two tracks.

Elements Specific to gx:MultiTrack

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore an altitude specification (for example, in the <gx:coord> element).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters.
- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an **absolute** altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the altitude as a value in meters above the sea floor. If the point is above land rather than sea, the altitude will be interpreted as being above the ground.
- **clampToSeaFloor** - The altitude specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<gx:interpolate>

Boolean value. If the multi-track contains multiple <gx:Track> elements, specifies whether to interpolate missing values between the end of the first track and the beginning of the next one. When the default value (0) is used, the icon or model stops at the end of one track and then jumps to the start of the next one.

Contains

- <gx:Track>

<NetworkLink>

Syntax

```
<NetworkLink id="ID">
  <!-- inherited from Feature element --><name>...</name>
<!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
   <atom:link href=" "/>              <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- specific to NetworkLink -->
  <refreshVisibility>0</refreshVisibility> <!-- boolean -->
  <flyToView>0</flyToView>                 <!-- boolean -->
  <Link>...</Link>
</NetworkLink>
```

Description

References a KML file or KMZ archive on a local or remote network. Use the <Link> element to specify the location of the KML file. Within that element, you can define the refresh options for updating the file, based on time and camera change. NetworkLinks can be used in combination with Regions to handle very large datasets efficiently.

Elements Specific to NetworkLink

<refreshVisibility>

Boolean value. A value of 0 leaves the visibility of features within the control of the Google Earth user. Set the value to 1 to reset the visibility of features each time the NetworkLink is refreshed. For example, suppose a Placemark within the linked KML file has <visibility> set to 1 and the NetworkLink has <refreshVisibility> set to 1. When the file is first loaded into Google Earth, the user can clear the check box next to the item to turn off display in the 3D viewer. However, when the NetworkLink is refreshed, the Placemark will be made visible again, since its original visibility state was TRUE.

<flyToView>

Boolean value. A value of 1 causes Google Earth to fly to the view of the LookAt or Camera in the NetworkLinkControl (if it exists). If the NetworkLinkControl does not contain an AbstractView element, Google Earth flies to the LookAt or Camera element in the Feature child within the <kml> element in the refreshed file. If the <kml> element does not have a LookAt or Camera specified, the view is unchanged. For example, Google Earth would fly to the <LookAt> view of the parent Document, not the <LookAt> of the Placemarks contained within the Document.

<Link> ***(required).*** **See <Link>.**

Tip: To display the top-level Folder or Document within a Network Link in the List View, assign an ID to the Folder or Document. Without this ID, only the child object names are displayed in the List View.

Example

```
<Document>
  <visibility>1</visibility>
  <NetworkLink>
    <name>NE US Radar</name>
    <refreshVisibility>1</refreshVisibility>
    <flyToView>1</flyToView>
    <Link>...</Link></NetworkLink>
</Document>
```

Extends

- *<Feature>*

Contained By

- any element derived from *<Container>*

<NetworkLinkControl>

Syntax

```
<NetworkLinkControl>
  <minRefreshPeriod>0</minRefreshPeriod>           <!-- float -->
  <maxSessionLength>-1</maxSessionLength>          <!-- float -->
  <cookie>...</cookie>                             <!-- string --
>
  <message>...</message>                           <!-- string -->
  <linkName>...</linkName>                         <!-- string --
>
  <linkDescription>...</linkDescription>           <!-- string --
>
  <linkSnippet maxLines="2">...</linkSnippet>      <!-- string --
>
  <expires>...</expires>                           <!-- kml:dateTime -->
  <Update>...</Update>                             <!-- Change,Create,Delete -
->
  <AbstractView>...</AbstractView>                 <!-- LookAt or Camera -->
</NetworkLinkControl>
```

Description

Controls the behavior of files fetched by a <NetworkLink>.

Elements Specific to NetworkLinkControl

<minRefreshPeriod>

Specified in seconds, <minRefreshPeriod> is the minimum allowed time between fetches of the file. This parameter allows servers to throttle fetches of a particular file and to tailor refresh rates to the expected rate of change to the data. For example, a user might set a link refresh to 5 seconds, but you could set your minimum refresh period to 3600 to limit refresh updates to once every hour.

<maxSessionLength>

Specified in seconds, <maxSessionLength> is the maximum amount of time for which the client NetworkLink can remain connected. The default value of -1 indicates not to terminate the session explicitly.

<cookie>

Use this element to append a string to the URL query on the next refresh of the network link. You can use this data in your script to provide more intelligent handling on the server side, including version querying and conditional file delivery.

<message>

You can deliver a pop-up message, such as usage guidelines for your network link. The message appears when the network link is first loaded into Google Earth, or when it is changed in the network link control.

<linkName>

You can control the name of the network link from the server, so that changes made to the name on the client side are overridden by the server.

<linkDescription>

You can control the description of the network link from the server, so that changes made to the description on the client side are overridden by the server.

<linkSnippet maxLines="2" >

You can control the snippet for the network link from the server, so that changes made to the snippet on the client side are overridden by the server. <linkSnippet> has a **maxLines** attribute, an integer that specifies the maximum number of lines to display.

<expires>

You can specify a date/time at which the link should be refreshed. This specification takes effect only if the <refreshMode> in <Link> has a value of **onExpire**. See <refreshMode>

<Update>

With <Update>, you can specify any number of Change, Create, and Delete tags for a *.kml* file or *.kmz* archive that has previously been loaded with a network link. See <Update>.

<AbstractView>

Example

```
<kml xmlns="http://www.opengis.net/kml/2.2">
<NetworkLinkControl>
   <message>This is a pop-up message. You will only see this once</message>
   <cookie>cookie=sometext</cookie>
   <linkName>New KML features</linkName>
   <linkDescription><![CDATA[KML now has new features available!]]
></linkDescription>
</NetworkLinkControl>
</kml>
```

Extends

- This is a direct child of the <kml> element.

Contained By

- <kml>

See Also

- <Update>

- <NetworkLink>

<Object>

Syntax

```
<!-- abstract element; do not create -->
<!-- Object id="ID" targetId="NCName" -->
<!-- /Object> -->
```

Description

This is an abstract base class and cannot be used directly in a KML file. It provides the **id** attribute, which allows unique identification of a KML element, and the **targetId** attribute, which is used to reference objects that have already been loaded into Google Earth. The **id** attribute must be assigned if the <Update> mechanism is to be used.

<Overlay>

Syntax

```
<!-- abstract element; do not create -->
<!-- Overlay id="ID" -->                    <!-- GroundOverlay,ScreenOverlay -
->
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>            <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- specific to Overlay -->
  <color>ffffffff</color>                   <!-- kml:color -->
  <drawOrder>0</drawOrder>                  <!-- int -->
  <Icon>
    <href>...</href>
  </Icon>
<!-- /Overlay -->
```

Description

This is an abstract element and cannot be used directly in a KML file. <*Overlay*> is the base type for image overlays drawn on the planet surface or on the screen. <Icon> specifies the image to use and can be configured to reload images based on a timer or by camera changes. This element also includes specifications for stacking order of multiple overlays and for adding color and transparency values to the base image.

Elements Specific to Overlay

<color>

Color values are expressed in hexadecimal notation, including opacity (alpha) values. The order of expression is alpha, blue, green, red (*aabbggrr*). The range of values for any one color is 0 to 255 (`00` to `ff`). For opacity, `00` is fully transparent and `ff` is fully opaque. For example, if you want to apply a blue color with 50 percent opacity to an overlay, you would specify the following: `<color>7fff0000</color>`

Note: The <geomColor> element has been deprecated. Use <color> instead.

<drawOrder>

This element defines the stacking order for the images in overlapping overlays. Overlays with higher <drawOrder> values are drawn on top of overlays with lower <drawOrder> values.

<Icon> See also <Icon>.

Defines the image associated with the Overlay. The <href> element defines the location of the image to be used as the Overlay. This location can be either on a local file system or on a web server. If this element is omitted or contains no <href>, a rectangle is drawn using the color and size defined by the ground or screen overlay.

```
<Icon>
   <href>icon.jpg</href>
</Icon>
```

Extends

- *<Feature>*

Extended By

- <GroundOverlay>
- <PhotoOverlay>
- <ScreenOverlay>

<PhotoOverlay>

Syntax

```
<PhotoOverlay>
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>            <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- inherited from Overlay element -->
  <color>ffffffff</color>               <!-- kml:color -->
  <drawOrder>0</drawOrder>              <!-- int -->
  <Icon>
    <href>...</href>                    <!-- anyURI -->
    ...
  </Icon>

  <!-- specific to PhotoOverlay -->
  <rotation>0</rotation>                <!-- kml:angle180 -->
  <ViewVolume>
    <leftFov>0</leftFov>                <!-- kml:angle180 -->
    <rightFov>0</rightFov>              <!-- kml:angle180 -->
    <bottomFov>0</bottomFov>            <!-- kml:angle90 -->
    <topFov>0</topFov>                 <!-- kml:angle90 -->
    <near>0</near>                     <!-- double -->
  </ViewVolume>
  <ImagePyramid>
    <tileSize>256</tileSize>           <!-- int -->
    <maxWidth>...</maxWidth>           <!-- int -->
    <maxHeight>...</maxHeight>         <!-- int -->
    <gridOrigin>lowerLeft</gridOrigin> <!-- lowerLeft or upperLeft -->
  </ImagePyramid>
  <Point>
    <coordinates>...</coordinates>     <!-- lon,lat[,alt] -->
  </Point>
  <shape>rectangle</shape>              <!-- kml:shape -->
</PhotoOverlay>
```

Description

The <PhotoOverlay> element allows you to geographically locate a photograph on the Earth and to specify viewing parameters for this PhotoOverlay. The PhotoOverlay can be a simple 2D rectangle, a partial or full cylinder, or a sphere (for spherical panoramas). The overlay is placed at the specified location and oriented toward the viewpoint.

Because <PhotoOverlay> is derived from <Feature>, it can contain one of the two elements derived from <AbstractView>—either <Camera> or <LookAt>. The Camera (or LookAt) specifies a *viewpoint* and a *viewing direction* (also referred to as a *view vector*). The PhotoOverlay is positioned in relation to the viewpoint. Specifically, the plane of a 2D rectangular image is orthogonal (at right angles to) the view vector. The normal of this plane—that is, its front, which is the part with the photo—is oriented toward the viewpoint.

The URL for the PhotoOverlay image is specified in the <Icon> tag, which is inherited from <Overlay>. The <Icon> tag must contain an <href> element that specifies the image file to use for the PhotoOverlay. In the case of a very large image, the <href> is a special URL that indexes into a pyramid of images of varying resolutions (see ImagePyramid).

For more information, see the "Topics in KML" page on PhotoOverlay.

Elements Specific to PhotoOverlay

<rotation>

Adjusts how the photo is placed inside the field of view. This element is useful if your photo has been rotated and deviates slightly from a desired horizontal view.

<ViewVolume>

Defines how much of the current scene is visible. Specifying the *field of view* is analogous to specifying the lens opening in a physical camera. A small field of view, like a telephoto lens, focuses on a small part of the scene. A large field of view, like a wide-angle lens, focuses on a large part of the scene.

<leftFov>

Angle, in degrees, between the camera's viewing direction and the left side of the view volume.

<rightFov>

Angle, in degrees, between the camera's viewing direction and the right side of the view volume.

<bottomFov>

Angle, in degrees, between the camera's viewing direction and the bottom side of the view volume.

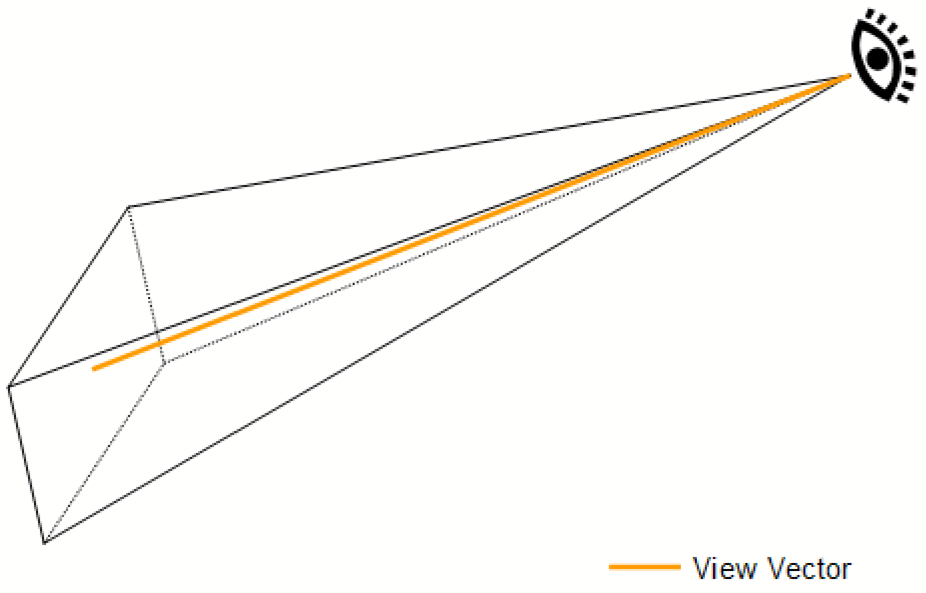
<topFov>

Angle, in degrees, between the camera's viewing direction and the top side of the view volume.

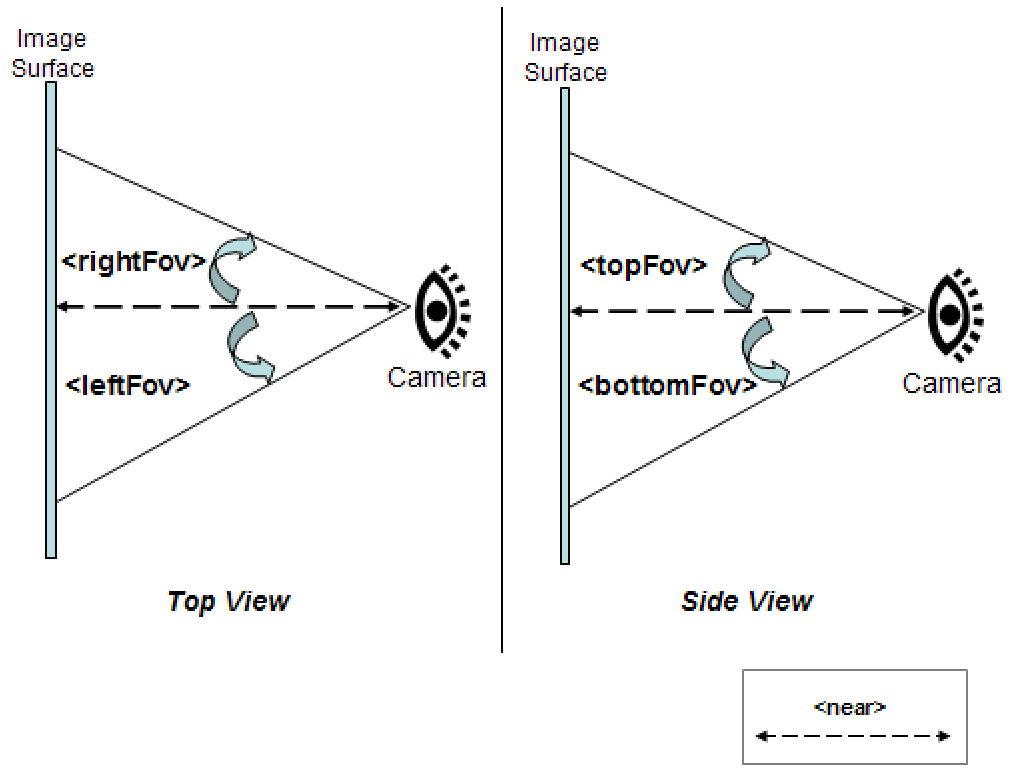
<near>

Measurement in meters along the viewing direction from the camera viewpoint to the PhotoOverlay shape.

The field of view for a PhotoOverlay is defined by four planes, each of which is specified by an angle relative to the *view vector*. These four planes define the top, bottom, left, and right sides of the field of view, which has the shape of a truncated pyramid, as shown here:



The following diagrams show the four field-of-view angles within this pyramid:



<ImagePyramid>

For very large images, you'll need to construct an image pyramid, which is a hierarchical set of images, each of which is an increasingly lower resolution version of the original image. Each image in the pyramid is subdivided into tiles, so that only the portions in view need to be loaded. Google Earth calculates the current viewpoint and loads the tiles that are appropriate to the user's distance from the image. As the viewpoint moves closer to the PhotoOverlay, Google Earth loads higher resolution tiles. Since all the pixels in the original image can't be viewed on the screen at once, this preprocessing allows Google Earth to achieve maximum performance because it loads only the portions of the image that are in view, and only the pixel details that can be discerned by the user at the current viewpoint.

When you specify an image pyramid, you also modify the <href> in the <Icon> element to include specifications for which tiles to load.

<tileSize>

Size of the tiles, in pixels. Tiles must be square, and <tileSize> must be a power of 2. A tile size of 256 (the default) or 512 is recommended. The original image is divided into tiles of this size, at varying resolutions.

<maxWidth>

Width in pixels of the original image.

<maxHeight>

Height in pixels of the original image.

<gridOrigin>

Specifies where to begin numbering the tiles in each layer of the pyramid. A value of `lowerLeft` specifies that row 1, column 1 of each layer is in the bottom left corner of the grid.

<Point>

The <Point> element acts as a <Point> inside a <Placemark> element. It draws an icon to mark the position of the PhotoOverlay. The icon drawn is specified by the <styleUrl> and <StyleSelector> fields, just as it is for <Placemark>.

<shape>

The PhotoOverlay is projected onto the <shape>. The <shape> can be one of the following:

rectangle (default) - for an ordinary photo

cylinder - for panoramas, which can be either partial or full cylinders

sphere - for spherical panoramas

Example

```
<PhotoOverlay>
  <!-- Feature elements -->
  <name>A simple non-pyramidal photo</name>
  <description>High above the ocean</description>
  <!-- Overlay elements -->
  <Icon>
  <!-- A simple normal jpeg image -->
  <href>small-photo.jpg</href>
  </Icon>
  <!-- PhotoOverlay elements -->
  <!-- default: <rotation> default is 0 -->
  <ViewVolume>
    <near>1000</near>
    <leftFov>-60</leftFov>
    <rightFov>60</rightFov>
    <bottomFov>-45</bottomFov>
    <topFov>45</topFov>
  </ViewVolume>
  <!-- if no ImagePyramid only level 0 is shown,
       fine for a non-pyramidal image -->
  <Point>
    <coordinates>1,1</coordinates>
  </Point>
  <!-- default: <shape> -->
</PhotoOverlay>
```

Extends

- <Overlay>

Contained By

- <Folder>, <Document>, or <kml>

<Placemark>

Syntax

```
<Placemark id="ID">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>                <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- specific to Placemark element -->
  <Geometry>...</Geometry>
</Placemark>
```

Description

A Placemark is a Feature with associated Geometry. In Google Earth, a Placemark appears as a list item in the Places panel. A Placemark with a Point has an icon associated with it that marks a point on the Earth in the 3D viewer. (In the Google Earth 3D viewer, a Point Placemark is the only object you can click or roll over. Other Geometry objects do not have an icon in the 3D viewer. To give the user something to click in the 3D viewer, you would need to create a MultiGeometry object that contains both a Point and the other Geometry object.)

Elements Specific to Placemark

- 0 or one *<Geometry>* elements

Example

```
<Placemark>
  <name>Google Earth - New Placemark</name>
  <description>Some Descriptive text.</description>
  <LookAt>
    <longitude>-90.86879847669974</longitude>
    <latitude>48.25330383601299</latitude>
    <range>440.8</range>
    <tilt>8.3</tilt>
    <heading>2.7</heading>
  </LookAt>
  <Point>
    <coordinates>-90.86948943473118,48.25450093195546,0</coordinates>
  </Point>
</Placemark>
```

Extends

- *<Feature>*

Contained By

- <Document>
- <Folder>

See Also

- <Icon>

<Point>

Syntax

```
<Point id="ID">
  <!-- specific to Point -->
  <extrude>0</extrude>                        <!-- boolean -->
  <altitudeMode>clampToGround</altitudeMode>
               <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or
absolute -->
        <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <coordinates>...</coordinates>              <!-- lon,lat[,alt] -->
</Point>
```

Description

A geographic location defined by longitude, latitude, and (optional) altitude. When a Point is contained by a Placemark, the point itself determines the position of the Placemark's name and icon. When a Point is extruded, it is connected to the ground with a line. This "tether" uses the current LineStyle.

Elements Specific to Point

<extrude>

Boolean value. Specifies whether to connect the point to the ground with a line. To extrude a Point, the value for <altitudeMode> must be either **relativeToGround**, **relativeToSeaFloor**, or **absolute**. The point is extruded toward the center of the Earth's sphere.

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore an altitude specification (for example, in the <coordinates> tag).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters. A typical use of this mode is for placing telephone poles or a ski lift.
- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an **absolute** altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters. A typical use of this mode is for aircraft placement.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the altitude as a value in meters above the sea floor. If the point is above land rather than sea, the altitude will be interpreted as being above the ground.
- **clampToSeaFloor** - The altitude specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<coordinates> *(required)*

A single tuple consisting of floating point values for longitude, latitude, and altitude (in that order). Longitude and latitude values are in degrees, where

- *longitude* ≥ −180 and <= 180
- *latitude* ≥ −90 and ≤ 90
- *altitude* values (optional) are in meters above sea level

Do not include spaces between the three values that describe a coordinate.

Example

```
<Point>
  <coordinates>-90.86948943473118,48.25450093195546</coordinates>
</Point>
```

Extends

- <Geometry>

Contained By

- <MultiGeometry>
- <Placemark>

<Polygon>

Syntax

```
<Polygon id="ID">
  <!-- specific to Polygon -->
  <extrude>0</extrude>                       <!-- boolean -->
  <tessellate>0</tessellate>                 <!-- boolean -->
  <altitudeMode>clampToGround</altitudeMode>
        <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or
absolute -->
        <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <outerBoundaryIs>
    <LinearRing>
      <coordinates>...</coordinates>         <!-- lon,lat[,alt] -->
    </LinearRing>
  </outerBoundaryIs>
  <innerBoundaryIs>
    <LinearRing>
      <coordinates>...</coordinates>         <!-- lon,lat[,alt] -->
    </LinearRing>
  </innerBoundaryIs>
</Polygon>
```

Description

A Polygon is defined by an outer boundary and 0 or more inner boundaries. The boundaries, in turn, 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.

The <coordinates> for polygons must be specified in counterclockwise order. 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.

Note: In Google Earth, a Polygon with an <altitudeMode> of *clampToGround* follows lines of constant bearing; however, a LinearRing (by itself) with an <altitudeMode> of *clampToGround* follows great circle lines.

Elements Specific to Polygon

<extrude>

Boolean value. Specifies whether to connect the Polygon to the ground. To extrude a Polygon, the altitude mode must be either **relativeToGround**, **relativeToSeaFloor**, or **absolute**. Only the vertices are extruded, not the geometry itself (for example, a rectangle turns into a box with five faces. The vertices of the Polygon are extruded toward the center of the Earth's sphere.

<tessellate>

This field is not used by Polygon. To allow a Polygon to follow the terrain (that is, to enable tessellation) specify an altitude mode of **clampToGround** or **clampToSeaFloor**.

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore an altitude specification (for example, in the <coordinates> tag).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the

elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters. A typical use of this mode is for placing telephone poles or a ski lift.

- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an **absolute** altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters. A typical use of this mode is for aircraft placement.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the altitude will be interpreted as being above the ground.
- **clampToSeaFloor** - The altitude specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<outerBoundaryIs> ***(required)***

Contains a <LinearRing> element.

<innerBoundaryIs>

Contains a <LinearRing> element. A Polygon can contain multiple <innerBoundaryIs> elements, which create multiple cut-outs inside the Polygon.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>Polygon.kml</name>
  <open>0</open>
  <Placemark>
    <name>hollow box</name>
    <Polygon>
      <extrude>1</extrude>
      <altitudeMode>relativeToGround</altitudeMode>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            -122.366278,37.818844,30
            -122.365248,37.819267,30
            -122.365640,37.819861,30
            -122.366669,37.819429,30
            -122.366278,37.818844,30
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
      <innerBoundaryIs>
        <LinearRing>
          <coordinates>
            -122.366212,37.818977,30
            -122.365424,37.819294,30
            -122.365704,37.819731,30
            -122.366488,37.819402,30
            -122.366212,37.818977,30
          </coordinates>
        </LinearRing>
      </innerBoundaryIs>
    </Polygon>
  </Placemark>
</Document>
</kml>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <Placemark>

<PolyStyle>

Syntax

```
<PolyStyle id="ID">
  <!-- inherited from ColorStyle -->
  <color>ffffffff</color>            <!-- kml:color -->
  <colorMode>normal</colorMode>      <!-- kml:colorModeEnum: normal or
random -->

  <!-- specific to PolyStyle -->
  <fill>1</fill>                     <!-- boolean -->
  <outline>1</outline>               <!-- boolean -->
</PolyStyle>
```

Description

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).

Elements Specific to PolyStyle

<fill>

Boolean value. Specifies whether to fill the polygon.

<outline>

Boolean value. Specifies whether to outline the polygon. Polygon outlines use the current LineStyle.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>PolygonStyle.kml</name>
  <open>1</open>
  <Style id="examplePolyStyle">
    <PolyStyle>
      <color>ff0000cc</color>
      <colorMode>random</colorMode>
    </PolyStyle>
  </Style>
  <Placemark>
    <name>hollow box</name>
    <styleUrl>#examplePolyStyle</styleUrl>
    <Polygon>
      <extrude>1</extrude>
      <altitudeMode>relativeToGround</altitudeMode>
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            -122.3662784465226,37.81884427772081,30
            -122.3652480684771,37.81926777010555,30
            -122.365640222455,37.81986126286519,30
            -122.36666937925,37.81942987753481,30
            -122.3662784465226,37.81884427772081,30
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
      <innerBoundaryIs>
```

```
        <LinearRing>
          <coordinates>
            -122.366212593918,37.81897719083808,30
            -122.3654241733188,37.81929450992014,30
            -122.3657048517827,37.81973175302663,30
            -122.3664882465854,37.81940249291773,30
            -122.366212593918,37.81897719083808,30
          </coordinates>
        </LinearRing>
      </innerBoundaryIs>
    </Polygon>
  </Placemark>
</Document>
</kml>
```

Extends

- *<ColorStyle>*

Contained By

- <Style>

<Region>

Syntax

```
<Region id="ID">
  <LatLonAltBox>
    <north></north>                            <!-- required; kml:angle90 -->
    <south></south>                            <!-- required; kml:angle90 -->
    <east></east>                              <!-- required; kml:angle180 -->
    <west></west>                              <!-- required; kml:angle180 -->
    <minAltitude>0</minAltitude>               <!-- float -->
    <maxAltitude>0</maxAltitude>               <!-- float -->
    <altitudeMode>clampToGround</altitudeMode>
        <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or
absolute -->
        <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  </LatLonAltBox>
  <Lod>
    <minLodPixels>0</minLodPixels>             <!-- float -->
    <maxLodPixels>-1</maxLodPixels>            <!-- float -->
    <minFadeExtent>0</minFadeExtent>           <!-- float -->
    <maxFadeExtent>0</maxFadeExtent>           <!-- float -->
  </Lod>
</Region>
```

Description

A region contains a *bounding box* (<LatLonAltBox>) that describes an area of interest defined by geographic coordinates and altitudes. In addition, a Region contains an *LOD (level of detail) extent* (<Lod>) that defines a validity range of the associated Region in terms of projected screen size. A Region is said to be "active" when the bounding box is within the user's view and the LOD requirements are met. Objects associated with a Region are drawn only when the Region is active. When the <viewRefreshMode> is **onRegion**, the Link or Icon is loaded only when the Region is active. See the "Topics in KML" page on Regions for more details. In a Container or NetworkLink hierarchy, this calculation uses the Region that is the closest ancestor in the hierarchy.

Elements Specific to Region

<LatLonAltBox>*(required)*

A bounding box that describes an area of interest defined by geographic coordinates and altitudes. Default values and required fields are as follows:

<altitudeMode> or <gx:altitudeMode>

Possible values for <altitudeMode> are **clampToGround**, **relativeToGround**, and **absolute**. Possible values for <gx:altitudeMode> are **clampToSeaFloor** and **relativeToSeaFloor**. Also see <LatLonBox>.

<minAltitude>

Specified in meters (and is affected by the altitude mode specification).

<maxAltitude>

Specified in meters (and is affected by the altitude mode specification).

<north> ***(required)***

Specifies the latitude of the north edge of the bounding box, in decimal degrees from 0 to ±90.

<south> ***(required)***

Specifies the latitude of the south edge of the bounding box, in decimal degrees from 0 to ±90.

<east> ***(required)***

Specifies the longitude of the east edge of the bounding box, in decimal degrees from 0 to ±180.

<west> ***(required)***

Specifies the longitude of the west edge of the bounding box, in decimal degrees from 0 to ±180.

```
<LatLonAltBox>
  <north>43.374</north>
  <south>42.983</south>
  <east>-0.335</east>
  <west>-1.423</west>
  <minAltitude>0</minAltitude>
  <maxAltitude>0</maxAltitude>
</LatLonAltBox>
```

<Lod>

Lod is an abbreviation for *Level of Detail*. <Lod> describes the size of the projected region on the screen that is required in order for the region to be considered "active." Also specifies the size of the pixel ramp used for fading in (from transparent to opaque) and fading out (from opaque to transparent). See diagram below for a visual representation of these parameters.

```
<Lod>
  <minLodPixels>256</minLodPixels>
  <maxLodPixels>-1</maxLodPixels>
  <minFadeExtent>0</minFadeExtent>
  <maxFadeExtent>0</maxFadeExtent>
</Lod>
```

<minLodPixels> ***(required)***

Measurement in screen pixels that represents the minimum limit of the visibility range for a given Region. Google Earth calculates the size of the Region when projected onto screen space. Then it computes the square root of the Region's area (if, for example, the Region is square and the viewpoint is directly above the Region, and the Region is not tilted, this measurement is equal to the width of the projected Region). If this measurement falls within the limits defined by <minLodPixels> and <maxLodPixels> (and if the <LatLonAltBox> is in view), the Region is active. If this limit is not reached, the associated geometry is considered to be too far from the user's viewpoint to be drawn.

<maxLodPixels>

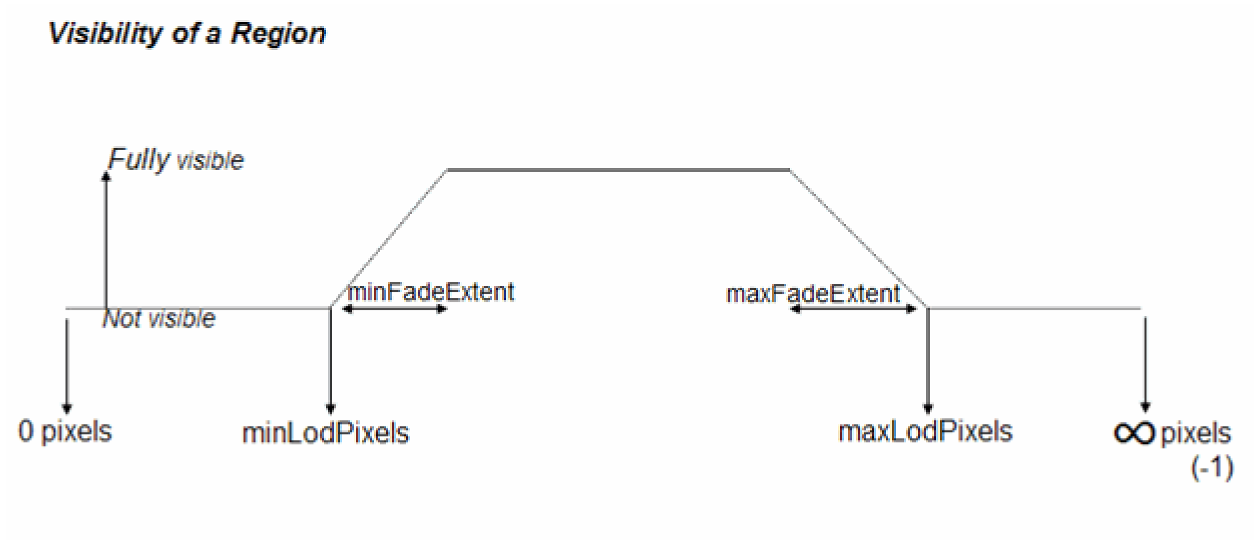
Measurement in screen pixels that represents the maximum limit of the visibility range for a given Region. A value of −1, the default, indicates "active to infinite size."

<minFadeExtent>

Distance over which the geometry fades, from fully opaque to fully transparent. This ramp value, expressed in screen pixels, is applied at the minimum end of the LOD (visibility) limits.

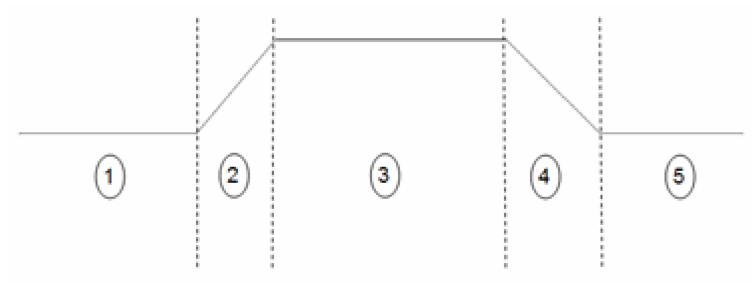
<maxFadeExtent>

Distance over which the geometry fades, from fully transparent to fully opaque. This ramp value, expressed in screen pixels, is applied at the maximum end of the LOD (visibility) limits.



In the following diagram, if *P*=the calculated projected pixel size, the circled numbers indicate the following:

```
if (P < minLodPixels)
  opacity=0                                 //#1 in diagram
else if(P < minLodPixels + minFadeExtent)
  opacity=(P - minLodPixels)/minFadeExtent  //#2 in diagram
else if (P < maxLodPixels - maxFadeExtent)
  opacity=1                                 //#3 in diagram
else if (P < maxLodPixels)
  opacity=(maxLodPixels-P)/maxFadeExtent    //#4 in diagram
else
  opacity=0                                 //#5 in diagram
```



Example of <Region>

```
<Region>
  <LatLonAltBox>
    <north>50.625</north>
    <south>45</south>
    <east>28.125</east>
    <west>22.5</west>
    <minAltitude>10</minAltitude>
    <maxAltitude>50</maxAltitude>
  </LatLonAltBox>
  <Lod>
    <minLodPixels>128</minLodPixels>
    <maxLodPixels>1024</maxLodPixels>
    <minFadeExtent>128</minFadeExtent>
    <maxFadeExtent>128</maxFadeExtent>
  </Lod>
</Region>
```

Extends

- *<Object>*

Contained By

- any element derived from *<Feature>*

<Schema>

Syntax

```
<Schema name="string" id="ID">
  <SimpleField type="string" name="string">
    <displayName>...</displayName>            <!-- string -->
  </SimpleField>
</Schema>
```

Description

Specifies a custom KML schema that is used to add custom data to KML Features. The "id" attribute is required and must be unique within the KML file. <Schema> is always a child of <Document>.

Elements Specific to Schema

A Schema element contains one or more SimpleField elements. In the SimpleField, the Schema declares the *type* and *name* of the custom field. It optionally specifies a *displayName* (the user-friendly form, with spaces and proper punctuation used for display in Google Earth) for this custom field.

<SimpleField type="*string*" name="*string*">

The declaration of the custom field, which must specify both the *type* and the *name* of this field. If either the *type* or the *name* is omitted, the field is ignored. The *type* can be one of the following:

- string
- int
- uint
- short
- ushort
- float
- double
- bool

<displayName>

The name, if any, to be used when the field name is displayed to the Google Earth user. Use the [CDATA] element to escape standard HTML markup.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <Schema name="TrailHeadType" id="TrailHeadTypeId">
    <SimpleField type="string" name="TrailHeadName">
      <displayName><![CDATA[<b>Trail Head Name</b>]]></displayName>
    </SimpleField>
    <SimpleField type="double" name="TrailLength">
      <displayName><![CDATA[<i>The length in miles</i>]]></displayName>
    </SimpleField>
    <SimpleField type="int" name="ElevationGain">
      <displayName><![CDATA[<i>change in altitude</i>]]></displayName>
    </SimpleField>
  </Schema>
</Document>
</kml>
```

Extends

This is a root element.

Contained By

- <Document>

See Also

- <SchemaData>

<ScreenOverlay>

Syntax

```
<ScreenOverlay id="ID">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author>...<atom:author>         <!-- xmlns:atom -->
  <atom:link href=" "/>               <!-- xmlns:atom -->
  <address>...</address>                <!-- string -->
  <xal:AddressDetails>...</xal:AddressDetails>  <!-- xmlns:xal -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- Camera or LookAt -->
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>              <!-- deprecated in KML 2.2 -->
  <ExtendedData>...</ExtendedData>      <!-- new in KML 2.2 -->

  <!-- inherited from Overlay element -->
  <color>ffffffff</color>                  <!-- kml:color -->
  <drawOrder>0</drawOrder>                 <!-- int -->
  <Icon>...</Icon>

  <!-- specific to ScreenOverlay -->
  <overlayXY x="double" y="double" xunits="fraction"
yunits="fraction"/>
    <!-- vec2 -->
    <!-- xunits and yunits can be one of: fraction, pixels, or
insetPixels -->
  <screenXY x="double" y="double" xunits="fraction"
yunits="fraction"/>
    <!-- vec2 -->
  <rotationXY x="double" y="double" xunits="fraction" yunits"fraction"/>
    <!-- vec2 -->
  <size x="double" y="double" xunits="fraction"
yunits="fraction"/>
    <!-- vec2 -->
  <rotation>0</rotation>                   <!-- float -->
 </ScreenOverlay>
```

Description

This element draws an image overlay fixed to the screen. Sample uses for ScreenOverlays are compasses, logos, and heads-up displays. ScreenOverlay sizing is determined by the <size> element. Positioning of the overlay is handled by mapping a point in the image specified by <overlayXY> to a point on the screen specified by <screenXY>. Then the image is rotated by <rotation> degrees about a point relative to the screen specified by <rotationXY>.

The <href> child of <Icon> specifies the image to be used as the overlay. This file can be either on a local file system or on a web server. If this element is omitted or contains no <href>, a rectangle is drawn using the color and size defined by the screen overlay.

Elements Specific to ScreenOverlay

<overlayXY>

Specifies a point on (or outside of) the overlay image that is mapped to the screen coordinate (<*screenXY*>). It requires *x* and *y* values, and the units for those values.

The *x* and *y* values can be specified in three different ways: as *pixels* (**"pixels"**), as *fractions* of the image (**"fraction"**), or as *inset pixels* (**"insetPixels"**), which is an offset in pixels from the upper right corner of the image. The *x* and *y* positions can be specified in different ways—for example, *x* can be in pixels and *y* can be a fraction. The origin of the coordinate system is in the lower left corner of the image.

- **x** - Either the number of pixels, a fractional component of the image, or a pixel inset indicating the *x* component of a point on the overlay image.
- **y** - Either the number of pixels, a fractional component of the image, or a pixel inset indicating the *y* component of a point on the overlay image.
- **xunits** - Units in which the *x* value is specified. A value of **"fraction"** indicates the *x* value is a fraction of the image. A value of **"pixels"** indicates the *x* value in pixels. A value of **"insetPixels"** indicates the indent from the right edge of the image.
- **yunits** - Units in which the *y* value is specified. A value of **"fraction"** indicates the *y* value is a fraction of the image. A value of **"pixels"** indicates the *y* value in pixels. A value of **"insetPixels"** indicates the indent from the top edge of the image.

<screenXY>

Specifies a point relative to the screen origin that the overlay image is mapped to. The *x* and *y* values can be specified in three different ways: as *pixels* (**"pixels"**), as *fractions* of the screen (**"fraction"**), or as *inset pixels* (**"insetPixels"**), which is an offset in pixels from the upper right corner of the screen. The *x* and *y* positions can be specified in different ways—for example, *x* can be in pixels and *y* can be a fraction. The origin of the coordinate system is in the lower left corner of the screen.

- **x** - Either the number of pixels, a fractional component of the screen, or a pixel inset indicating the *x* component of a point on the screen.
- **y** - Either the number of pixels, a fractional component of the screen, or a pixel inset indicating the *y* component of a point on the screen.
- **xunits** - Units in which the *x* value is specified. A value of **"fraction"** indicates the *x* value is a fraction of the screen. A value of **"pixels"** indicates the *x* value in pixels. A value of **"insetPixels"** indicates the indent from the right edge of the screen.
- **yunits** - Units in which the *y* value is specified. A value of *fraction* indicates the *y* value is a fraction of the screen. A value of **"pixels"** indicates the *y* value in pixels. A value of **"insetPixels"** indicates the indent from the top edge of the screen.

For example, <*screenXY* x=".5" y=".5" xunits="fraction" yunits="fraction"/> indicates a point in the middle of the screen.

Here are some examples:

Center the image:

```
<ScreenOverlay>
  <overlayXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
  <screenXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
</ScreenOverlay>
```

Place the image on the top left:

```
<ScreenOverlay>
  <overlayXY x="0" y="1" xunits="fraction" yunits="fraction"/>
  <screenXY x="0" y="1" xunits="fraction" yunits="fraction"/>
</ScreenOverlay>
```

Place the image at the right of the screen:

```
<ScreenOverlay>
  <overlayXY x="1" y="1" xunits="fraction" yunits="fraction"/>
  <screenXY x="1" y="1" xunits="fraction" yunits="fraction"/>
```

```
</ScreenOverlay>
```

<rotationXY>

Point relative to the screen about which the screen overlay is rotated.

<size>

Specifies the size of the image for the screen overlay, as follows:

- A value of −1 indicates to use the native dimension
- A value of 0 indicates to maintain the aspect ratio
- A value of *n* sets the value of the dimension

For example:

To force the image to retain its original *x* and *y* dimensions, set the values to −1:

```
<size x="-1" y="-1" xunits="fraction" yunits="fraction"/>
```

To force the image to retain its horizontal dimension, but to take up 20 percent of the vertical screen space:

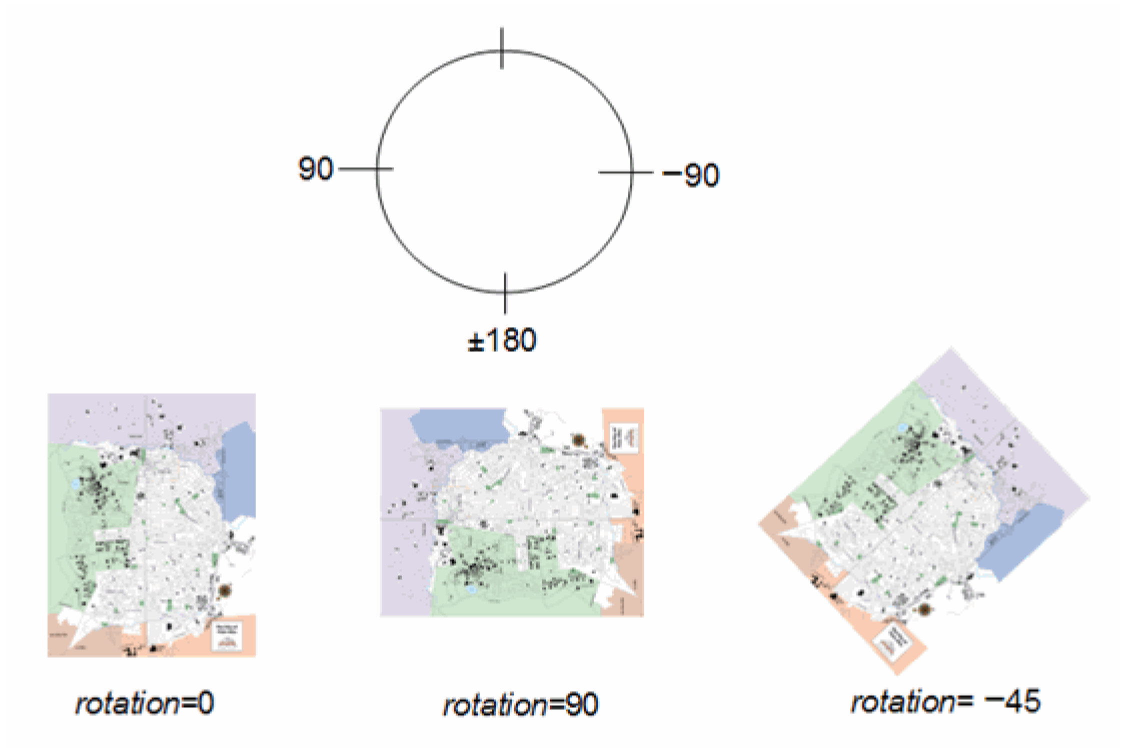
```
<size x="-1" y="0.2" xunits="fraction" yunits="fraction"/>
```

To force the image to resize to 100px by 500px:

```
<size x="100" y="500" xunits="pixels" yunits="pixels"/>
```

<rotation>

Indicates the angle of rotation of the parent object. A value of 0 means no rotation. The value is an angle in degrees counterclockwise starting from north. Use ±180 to indicate the rotation of the parent object from 0. The center of the <rotation>, if not (.5,.5), is specified in <rotationXY>.



Example

The following example places an image at the exact center of the screen, using the original width, height, and aspect ratio of the image.

```
<ScreenOverlay id="khScreenOverlay756">
  <name>Simple crosshairs</name>
  <description>This screen overlay uses fractional positioning
   to put the image in the exact center of the screen</description>
  <Icon>
    <href>http://myserver/myimage.jpg</href>
  </Icon>
  <overlayXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
  <screenXY x="0.5" y="0.5" xunits="fraction" yunits="fraction"/>
  <rotation>39.37878630116985</rotation>
  <size x="0" y="0" xunits="pixels" yunits="pixels"/>
</ScreenOverlay>
```

Extends

- *<Feature>*
- *<Overlay>*

Contained By

- <Document>
- <Folder>

<Style>

Syntax

```
<Style id="ID">
<!-- extends StyleSelector -->

<!-- specific to Style -->
  <IconStyle>...</IconStyle>
  <LabelStyle>...</LabelStyle>
  <LineStyle>...</LineStyle>
  <PolyStyle>...</PolyStyle>
  <BalloonStyle>...</BalloonStyle>
  <ListStyle>...</ListStyle>
</Style>
```

Description

A Style defines an addressable style group that can be referenced by StyleMaps and Features. Styles affect how Geometry is presented in the 3D viewer and how Features appear in the Places panel of the List view. Shared styles are collected in a <Document> and must have an **id** defined for them so that they can be referenced by the individual Features that use them.

Use an **id** to refer to the style from a <styleUrl>.

Example

```
<Document>
  <!-- Begin Style Definitions -->
  <Style id="myDefaultStyles">
    <IconStyle>
      <color>a1ff00ff</color>
      <scale>1.399999976158142</scale>
      <Icon>
        <href>http://myserver.com/icon.jpg</href>
      </Icon>
    </IconStyle>
    <LabelStyle>
      <color>7fffaaff</color>
      <scale>1.5</scale>
    </LabelStyle>
    <LineStyle>
      <color>ff0000ff</color>
      <width>15</width>
    </LineStyle>
    <PolyStyle>
      <color>7f7faaaa</color>
      <colorMode>random</colorMode>
    </PolyStyle>
  </Style>
  <!-- End Style Definitions -->
  <!-- Placemark #1 -->
  <Placemark>
    <name>Google Earth - New Polygon</name>
    <description>Here is some descriptive text</description>
    <styleUrl>#myDefaultStyles</styleUrl>
    . . .
  </Placemark>
  <!-- Placemark #2 -->
  <Placemark>
    <name>Google Earth - New Path</name>
    <styleUrl>#myDefaultStyles</styleUrl>
      . . . .
  </Placemark>
</Document>
</kml>
```

Extends

- *<StyleSelector>*

Contained By

- any *<Feature>*

Elements Specific to Style

- <BalloonStyle>
- <IconStyle>
- <LabelStyle>
- <LineStyle>
- <ListStyle>
- <PolyStyle>

<StyleMap>

Syntax

```
<StyleMap id="ID">
  <!-- extends StyleSelector -->
  <!-- elements specific to StyleMap -->
  <Pair id="ID">
    <key>normal</key>              <!-- kml:styleStateEnum:  normal or
highlight -->
    <styleUrl>...</styleUrl> or <Style>...</Style>
  </Pair>
</StyleMap>
```

Description

A <StyleMap> maps between two different Styles. Typically a <StyleMap> element is used to provide separate normal and highlighted styles for a placemark, so that the highlighted version appears when the user mouses over the icon in Google Earth.

Elements Specific to StyleMap

<Pair> *(required)*

Defines a key/value pair that maps a mode (*normal* or *highlight*) to the predefined <styleUrl>. <Pair> contains two elements (both are required):

- **<key>**, which identifies the key
- **<styleUrl>** or **<Style>**, which references the style. In <styleUrl>, for referenced style elements that are local to the KML document, a simple # referencing is used. For styles that are contained in external files, use a full URL along with # referencing. For example:

```
<Pair>
  <key>normal</key>

<styleUrl>http://myserver.com/populationProject.xml#example_style_off</st
yleUrl>
</Pair>
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <name>StyleMap.kml</name>
  <open>1</open>
  <Style id="normalState">
    <IconStyle>
      <scale>1.0</scale>
      <Icon>
        <href>http://maps.google.com/mapfiles/kml/pal3/icon55.png</href>
      </Icon>
    </IconStyle>
    <LabelStyle>
      <scale>1.0</scale>
    </LabelStyle>
  </Style>
  <Style id="highlightState">
    <IconStyle>
      <Icon>
        <href>http://maps.google.com/mapfiles/kml/pal3/icon60.png</href>
      </Icon>
      <scale>1.1</scale>
    </IconStyle>
    <LabelStyle>
      <scale>1.1</scale>
      <color>ff0000c0</color>
    </LabelStyle>
  </Style>
  <StyleMap id="styleMapExample">
    <Pair>
      <key>normal</key>
      <styleUrl>#normalState</styleUrl>
    </Pair>
    <Pair>
      <key>highlight</key>
      <styleUrl>#highlightState</styleUrl>
    </Pair>
  </StyleMap>
  <Placemark>
    <name>StyleMap example</name>
    <styleUrl>#styleMapExample</styleUrl>
    <Point>
      <coordinates>-122.368987,37.817634,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

Extends

- *<StyleSelector>*

Contained By

- any *<Feature>*

<StyleSelector>

Syntax

```
<!-- abstract element; do not create -->
<!-- StyleSelector id="ID" -->                <!-- Style,StyleMap -->
<!-- /StyleSelector -->
```

Description

This is an abstract element and cannot be used directly in a KML file. It is the base type for the <Style> and <StyleMap> elements. The StyleMap element selects a style based on the current mode of the Placemark. An element derived from StyleSelector is uniquely identified by its **id** and its *url*.

Elements Specific to StyleSelector

This abstract element does not contain any child elements.

Extends

- *<Object>*

Extended By

- <Style>
- <StyleMap>

<TimePrimitive>

Syntax

```
<!-- abstract element; do not create -->
<!-- TimePrimitive id="ID" -->            <!-- TimeSpan,TimeStamp --
>
  <!-- extends Object -->
<!-- /TimePrimitive -->
```

Description

This is an abstract element and cannot be used directly in a KML file. This element is extended by the <TimeSpan> and <TimeStamp> elements.

Extends

- *<Object>*

Extended By

- <TimeSpan>
- <TimeStamp>

<TimeSpan>

Syntax

```
<TimeSpan id="ID">
  <begin>...</begin>     <!-- kml:dateTime -->
  <end>...</end>         <!-- kml:dateTime -->
</TimeSpan>
```

Description

Represents an extent in time bounded by begin and end *dateTimes*.

If <begin> or <end> is missing, then that end of the period is unbounded (see Example below).

The *dateTime* is defined according to XML Schema time (see XML Schema Part 2: Datatypes Second Edition). The value can be expressed as *yyyy-mm-dd*T*hh:mm:ss*zzzzzz, where T is the separator between the date and the time, and the time zone is either Z (for UTC) or *zzzzzz*, which represents ±*hh:mm* in relation to UTC. Additionally, the value can be expressed as a date only. See <TimeStamp> for examples.

Elements Specific to TimeSpan

<begin>

Describes the beginning instant of a time period. If absent, the beginning of the period is unbounded.

<end>

Describes the ending instant of a time period. If absent, the end of the period is unbounded.

Example

The following example shows the time period representing Colorado's statehood. It contains only a <begin> tag because Colorado became a state on August 1, 1876, and continues to be a state:

```
<Placemark>
  <name>Colorado</name>
  .
  .
  .
  <TimeSpan>
    <begin>1876-08-01</begin>
  </TimeSpan>
</Placemark>
```

Extends

- *<TimePrimitive>*

Contained By

- any element derived from *<Feature>*

<TimeStamp>

Syntax

```
<TimeStamp id=ID>
  <when>...</when>      <!-- kml:dateTime -->
</TimeStamp>
```

Description

Represents a single moment in time. This is a simple element and contains no children. Its value is a *dateTime*, specified in XML time (see XML Schema Part 2: Datatypes Second Edition). The precision of the TimeStamp is dictated by the *dateTime* value in the <when> element.

Elements Specific to TimeStamp

<when>

Specifies a single moment in time. The value is a *dateTime*, which can be one of the following:

- *dateTime* gives second resolution
- *date* gives day resolution
- *gYearMonth* gives month resolution
- *gYear* gives year resolution

The following examples show different resolutions for the <when> value:

- *gYear* (*YYYY*)

```
<TimeStamp>
  <when>1997</when>
</TimeStamp>
```

- *gYearMonth* (*YYYY-MM*)

```
<TimeStamp>
  <when>1997-07</when>
</TimeStamp>
```

- *date* (*YYYY-MM-DD*)

```
<TimeStamp>
  <when>1997-07-16</when>
</TimeStamp>
```

- *dateTime* (*YYYY-MM-DD***T***hh:mm:ss***Z**)
  Here, T is the separator between the calendar and the hourly notation of time, and Z indicates UTC. (Seconds are required.)

```
<TimeStamp>
  <when>1997-07-16T07:30:15Z</when>
</TimeStamp>
```

- *dateTime* (*YYYY-MM-DD***T***hh:mm:sszzzzzz)*
  This example gives the local time and then the ± conversion to UTC.

```
<TimeStamp>
  <when>1997-07-16T10:30:15+03:00</when>
</TimeStamp>
```

Extends

- *<TimePrimitive>*

Contained By

- any element that extends *<Feature>*

<gx:TimeSpan> and <gx:TimeStamp>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

A copy of the `<TimeSpan>` and `<TimeStamp>` elements, in the extension namespace. This allows for the inclusion of time values in AbstractViews (`<Camera>` and `<LookAt>`). Time values are used to control historical imagery, sunlight, and visibility of time-stamped Features.

Example

```
<?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">

  <Document>
    <name>Views with Time</name>
    <open>1</open>
    <description>
      In Google Earth, enable historical imagery and sunlight,
      then click on each placemark to fly to that point in
time.
    </description>

    <Placemark>
      <name>Sutro Baths in 1946</name>
      <Camera>
        <gx:TimeStamp>
          <when>1946-07-29T05:00:00-08:00</when>
        </gx:TimeStamp>
        <longitude>-122.518172</longitude>
        <latitude>37.778036</latitude>
        <altitude>221.0</altitude>
        <heading>70.0</heading>
        <tilt>75.0</tilt>
      </Camera>
    </Placemark>

    <Placemark>
      <name>Palace of Fine Arts in 2002</name>
      <Camera>
        <gx:TimeStamp>
          <when>2002-07-09T19:00:00-08:00</when>
        </gx:TimeStamp>
        <longitude>-122.444633</longitude>
        <latitude>37.801899</latitude>
        <altitude>139.629438</altitude>
        <heading>-70.0</heading>
        <tilt>75</tilt>
      </Camera>
    </Placemark>

  </Document>
</kml>
```

<gx:Tour>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:Tour>
  <name>...</name>
  <description>...</description>
  <gx:Playlist>

    <!-- any number of gx:TourPrimitive elements -->

  </gx:Playlist>
</gx:Tour>
```

Description

`<gx:Tour>` can contain a single `<gx:Playlist>` element, which in turn contains an ordered list of `gx:TourPrimitive` elements that define a tour in any KML browser. Learn more about tours.

Example

A number of tour examples are available from the Touring chapter of the **KML Developer's Guide**.

Contains

- **gx:Playlist** - Contains any number of `gx:TourPrimitive` elements. There can be zero or one `<gx:Playlist>` elements contained within a `<gx:Tour>` element.

```
<gx:Tour>
  <gx:Playlist>
    <!-- gx:TourPrimitive -->
      ...
    <!-- /gx:TourPrimitive -->

    <!--- Any number of gx:TourPrimitive elements can be included --
->
  </gx:Playlist>
</gx:Tour>
```

<gx:TourPrimitive>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.0 and later. Learn more

Syntax

```
<gx:Tour>
  <gx:Playlist>

    <!-- abstract element; do not create -->
    <!-- gx:TourPrimitive -->    <!-- gx:AnimatedUpdate, gx:FlyTo,
gx:TourControl, gx:SoundCue, gx:Wait -->
        <!-- extends Object -->
    <!-- /gx:TourPrimitive -->

  </gx:Playlist>
</gx:Tour>
```

Description

This is an abstract element and cannot be used directly in a KML file. This element is extended by the `<gx:FlyTo>`, `<gx:AnimatedUpdate>`, `<gx:TourControl>`, `<gx:Wait>`, and `<gx:SoundCue>` elements.

Elements extended from `gx:TourPrimitive` provide instructions to KML browsers during tours, including points to fly to and the duration of those flights, pauses, updates to KML features, and sound files to play.

These elements must be contained within a `<gx:Playlist>` element, which in turn is contained with a `<gx:Tour>` element.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">

  <Document>
    <name>gx:AnimatedUpdate example</name>
    <open>1</open>

    <Style>
      <IconStyle id="iconstyle">
        <scale>1.0</scale>
      </IconStyle>
    </Style>

    <Placemark id="mountainpin1">
      <name>New Zealand's Southern Alps</name>
      <styleUrl>#style</styleUrl>
      <Point>
        <coordinates>170.144,-43.605,0</coordinates>
      </Point>
    </Placemark>

    <gx:Tour>
      <name>Play me!</name>
      <gx:Playlist>

        <!-- The order and duration of TourPrimitives is important;
             in this example, the AnimatedUpdate needs
             6.5 seconds to complete. The FlyTo provides 4.1,
             and the Wait 2.4, giving the update time to
             complete before the Tour ends. AnimatedUpdates
             don't hold Tours open, but FlyTos and Waits do.

             For more information, refer to:

http://code.google.com/apis/kml/documentation/touring.html#tourtimelin
es
        -->
```

Contained by

- <gx:Playlist>

Extended by

- <gx:AnimatedUpdate>
- <gx:FlyTo>
- **<gx:SoundCue>**

```
<gx:SoundCue>
  <href>http://www.example.com/audio/trumpets.mp3</href>   <!-- any
URI -->
  <gx:delayedStart>0</gx:delayedStart>                     <!--
double -->
</gx:SoundCue>
```

  Contains an `<href>` element specifying a sound file to play, in MP3, M4A, or AAC format. It does not contain a duration. The sound file plays in parallel to the rest of the tour, meaning that the next tour primitive takes place immediately after the `<gx:SoundCue>` tour primitive is reached. If another sound file is cued before the first has finished playing, the files are mixed. The `<gx:delayedStart>` element specifies to delay the start of the sound for a given number of seconds before playing the file.

- **<gx:TourControl>**

```
<gx:TourControl>
  <gx:playMode>pause</gx:playMode>    <!-- gx:playModeEnum: pause -->
</gx:TourControl>
```

Contains a single `<gx:playMode>` element, allowing the tour to be paused until a user takes action to continue the tour.

- **<gx:Wait>**

```
<gx:Wait>
  <gx:duration>0.0</gx:duration>    <!-- double -->
</gx:Wait>
```

The camera remains still, at the last-defined gx:AbstractView, for the number of seconds specified before playing the next `gx:TourPrimitive`. Note that a wait does not pause the tour timeline - currently-playing sound files and animated updates will continue to play while the camera is waiting.

Extends

- <Object>

<gx:Track>

This element is an extension of the OGC KML 2.2 standard and is supported in Google Earth 5.2 and later. Learn more

Syntax

```
<gx:Track id="ID">
  <!-- specific to Track -->
  <altitudeMode>clampToGround</altitudeMode>
      <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or
absolute -->
      <!-- or, substitute gx:altitudeMode: clampToSeaFloor,
relativeToSeaFloor -->
  <when>...</when>                         <!-- kml:dateTime -->
  <gx:coord>...</gx:coord>                 <!-- string -->
  <gx:angles>...</gx:angles>               <!-- string -->
  <Model>...</Model>
  <ExtendedData>
    <SchemaData schemaUrl="anyURI">
      <gx:SimpleArrayData kml:name="string">
        <gx:value>...</gx:value>            <!-- string -->
      </gx:SimpleArrayData>
    <SchemaData>
  </ExtendedData>
</gx:Track>
```

Description

A *track* describes how an object moves through the world over a given time period. This feature allows you to create one visible object in Google Earth (either a Point icon or a Model) that encodes multiple positions for the same object for multiple times. In Google Earth, the time slider allows the user to move the view through time, which animates the position of the object.

A gx:MultiTrack element is used to collect multiple tracks into one conceptual unit with one associated icon (or Model) that moves along the track. This feature is useful if you have multiple tracks for the same real-world object. The <gx:interpolate> Boolean element of a <gx:MultiTrack> specifies whether to interpolate between the tracks in a multi-track. If this value is 0, then the point or Model stops at the end of one track and jumps to the start of the next one. (For example, if you want a single placemark to represent your travels on two days, and your GPS unit was turned off for four hours during this period, you would want to show a discontinuity between the points where the unit was turned off and then on again.) If the value for <gx:interpolate> is 1, the values between the end of the first track and the beginning of the next track are interpolated so that the track appears as one continuous path.

See the *Google Earth User Guide* for information on how to import GPS data into Google Earth.

Why are tracks useful?
Earlier versions of KML (pre–Google Earth 5.2) allow you to associate a time element with any Feature (placemark, ground overlay, etc.). However, you could only associate *one* time element with a given Feature. Tracks are a more efficient mechanism for associating time data with visible Features, since you create only one Feature, which can be associated with multiple time elements as the object moves through space.

In addition, the track element is more powerful than the earlier mechanism (described in the Time and Animation chapter of the ***KML Developer's Guide***) because <Track> provides a mechanism for interpolating the position of the object at any time along its track. With this new feature, Google Earth displays a graph of elevation and speed profiles (plus custom data, if present) for the object over time.

"Sparse" Data
When some data values are missing for positions on the track, empty <coord/> (<coord></coord>) or <angles/> (<angles></angles>) tags can be provided to balance the arrays. An empty <coord/> or <angles/> tag indicates that no such data exists for a given data point, and the value should be interpolated between the nearest two well-specified data points. This behavior also applies to ExtendedData for a track. Any element except <when> can be empty and will be interpolated between the nearest two well-specified elements.

Elements Specific to Track

<altitudeMode>

Specifies how *altitude* components in the <coordinates> element are interpreted. Possible values are

- **clampToGround** - (default) Indicates to ignore an altitude specification (for example, in the <gx:coord> tag).
- **relativeToGround** - Sets the altitude of the element relative to the actual ground elevation of a particular location. For example, if the ground elevation of a location is exactly at sea level and the altitude for a point is set to 9 meters, then the elevation for the icon of a point placemark elevation is 9 meters with this mode. However, if the same coordinate is set over a location where the ground elevation is 10 meters above sea level, then the elevation of the coordinate is 19 meters.
- **absolute** - Sets the altitude of the coordinate relative to sea level, regardless of the actual elevation of the terrain beneath the element. For example, if you set the altitude of a coordinate to 10 meters with an **absolute** altitude mode, the icon of a point placemark will appear to be at ground level if the terrain beneath is also 10 meters above sea level. If the terrain is 3 meters above sea level, the placemark will appear elevated above the terrain by 7 meters.

<gx:altitudeMode>

A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:

- **relativeToSeaFloor** - Interprets the altitude as a value in meters above the sea floor. If the point is above land rather than sea, the altitude will be interpreted as being above the ground.
- **clampToSeaFloor** - The altitude specification is ignored, and the point will be positioned on the sea floor. If the point is on land rather than at sea, the point will be positioned on the ground.

<when>

A *time* value that corresponds to a *position* (specified in a <gx:coord> element). The number of <when> elements in a <Track> must be equal to the number of <gx:coord> elements (and <gx:angles> elements, if included). You can specify an empty <when> element for a missing value if necessary.

<gx:coord>

A coordinate value consisting of three values for longitude, latitude, and altitude, with no comma separators. For example:

```
<gx:coord>-122.207881 37.371915 156.000000</gx:coord>
```

Note that the syntax for the <gx:coord> element is different from the syntax for the <coordinates> element, which uses commas to separate the longitude, latitude, and altitude components.

<gx:angles>

This value is used to specify an additional heading, tilt, and roll value to the icon or model for each time/position within the track. The three floating point values are listed without comma separators and represent degrees of rotation. If <gx:angles> is not specified, then Google Earth infers the heading, tilt, and roll of the object from its track.The number of <gx:angles> elements specified should equal the number of time (<when>) and position (<gx:coord>) elements.

Currently, icons support only heading, but models support all three values.

Here is an example of setting this value:

```
<gx:angles>45.54676 66.2342 77.0</gx:angles>
```

<Model>

If specified, the Model replaces the Point icon used to indicate the current position on the track. When a <Model> is specified within a <gx:Track>, here is how the child elements of <Model> function:

- The <Location> element is ignored.
- The <altitudeMode> element is ignored.
- The <Oriention> value is combined with the orientation of the track as follows. First, the <Orientation> rotation is applied, which brings the model from its local (x, y, z) coordinate system to a right-side-up, north-facing orientation. Next, a rotation is applied that corresponds to the interpolation of the <gx:angles> values that affect the heading, tilt, and roll of the model as it moves along the track. If no angles are specified, the heading and tilt are inferred from the movement of the model.

  Tip: If you are unsure of how to specify the orientation, omit the <Orientation> element from the <Model> and watch how Google Earth positions the model as it moves along the track. If you notice that the front of the model is facing sideways, modify the <heading> element in <Orientation> to rotate the model so that it points toward the front. If the model is not upright, try modifying the <tilt> or <roll> elements.

<ExtendedData>

Custom data elements defined in a <Schema> earlier in the KML file.

It's often useful to add extended data associated with each time/position on a track. Bicycle rides, for example, could include data for heart rate, cadence, and power, as shown in Example of Track with Extended Data. In the <Schema>, you define a <gx:SimpleArrayField> for each custom data type. Then, for each data type, include a <gx:SimpleArrayData> element containing <gx:value> elements that correspond to each time/position on the track. See the Adding Custom Data chapter of the ***KML Developer's Guide*** for more information on adding new data fields. In Google Earth, custom data is displayed in the Elevation Profile for the track.

Simple Example

This very basic example shows how to create parallel "arrays" of values for <when> and <gx:coord>. The number of time and position values must be equal.

```
<?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">
<Folder>
  <Placemark>
    <gx:Track>
      <when>2010-05-28T02:02:09Z</when>
      <when>2010-05-28T02:02:35Z</when>
      <when>2010-05-28T02:02:44Z</when>
      <when>2010-05-28T02:02:53Z</when>
      <when>2010-05-28T02:02:54Z</when>
      <when>2010-05-28T02:02:55Z</when>
      <when>2010-05-28T02:02:56Z</when>
      <gx:coord>-122.207881 37.371915 156.000000</gx:coord>
      <gx:coord>-122.205712 37.373288 152.000000</gx:coord>
      <gx:coord>-122.204678 37.373939 147.000000</gx:coord>
      <gx:coord>-122.203572 37.374630 142.199997</gx:coord>
      <gx:coord>-122.203451 37.374706 141.800003</gx:coord>
      <gx:coord>-122.203329 37.374780 141.199997</gx:coord>
      <gx:coord>-122.203207 37.374857 140.199997</gx:coord>
    </gx:Track>
  </Placemark>
</Folder>
```

```
</kml>
```

Example of Track with Extended Data

The boldface type in this example highlights the elements used to define and specify custom data for a bike ride. The custom data fields are internally named "heartrate," "cadence," and "power." The <Schema> element defines the name to display for each set of values (Heart Rate, Cadence, and Power) and specifies the data type for each new field (heartrate and cadence are of type `int`, and power is of type `float`). In Google Earth, this custom data is shown with the elevation profile for the track.

This example is a more realistic presentation of a track, with custom icons and separate icon and line styles for highlight and normal modes. Note, however, that the example includes only seven sets of data values. The actual example includes tens of thousands of values. (Data courtesy of Sean Broeder. This data was collected with a Garmin Edge 705 with associated heart rate monitor and power meter.)

```
<?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">
  <Document>
    <name>GPS device</name>
    <Snippet>Created Wed Jun 2 15:33:39 2010</Snippet>

    <!-- Normal track style -->
    <LookAt>
      <gx:TimeSpan>
        <begin>2010-05-28T02:02:09Z</begin>
        <end>2010-05-28T02:02:56Z</end>
      </gx:TimeSpan>
      <longitude>-122.205544</longitude>
      <latitude>37.373386</latitude>
      <range>1300.000000</range>
    </LookAt>
    <Style id="track_n">
      <IconStyle>
        <scale>.5</scale>
        <Icon>
          <href>http://earth.google.com/images/kml-icons/track-
directional/track-none.png</href>
        </Icon>
      </IconStyle>
      <LabelStyle>
        <scale>0</scale>
      </LabelStyle>

    </Style>
    <!-- Highlighted track style -->
    <Style id="track_h">
      <IconStyle>
        <scale>1.2</scale>
        <Icon>
          <href>http://earth.google.com/images/kml-icons/track-
directional/track-none.png</href>
        </Icon>
```

Extends

- *<Geometry>*

Contained By

- <MultiGeometry>
- <gx:MultiTrack>
- <Placemark>

<Update>

Syntax

```
<Update>
  <targetHref>...</targetHref>    <!-- URL -->
  <Change>...</Change>
  <Create>...</Create>
  <Delete>...</Delete>
</Update>
```

Description

Specifies an addition, change, or deletion to KML data that has already been loaded using the specified URL. The <targetHref> specifies the *.kml* or *.kmz* file whose data (within Google Earth) is to be modified. <Update> is always contained in a NetworkLinkControl. Furthermore, the file containing the NetworkLinkControl must have been loaded by a NetworkLink. See the "Topics in KML" page on Updates for a detailed example of how Update works.

Elements Specific to Update

Can contain any number of <Change>, <Create>, and <Delete> elements, which will be processed in order.

<targetHref> ***(required)***

A URL that specifies the *.kml* or *.kmz* file whose data (within Google Earth) is to be modified by an <Update> element. This KML file must already have been loaded via a <NetworkLink>. In that file, the element to be modified must already have an explicit **id** attribute defined for it.

<Change>

Modifies the values in an element that has already been loaded with a <NetworkLink>. Within the Change element, the child to be modified must include a **targetId** attribute that references the original element's **id**.

This update can be considered a "sparse update": in the modified element, only the values listed in <Change> are replaced; all other values remained untouched. When <Change> is applied to a set of coordinates, the new coordinates replace the current coordinates.

Children of this element are the element(s) to be modified, which are identified by the **targetId** attribute.

<Create>

Adds new elements to a Folder or Document that has already been loaded via a <NetworkLink>. The <targetHref> element in <Update> specifies the URL of the *.kml* or *.kmz* file that contained the original Folder or Document. Within that file, the Folder or Document that is to contain the new data must already have an explicit **id** defined for it. This **id** is referenced as the **targetId** attribute of the Folder or Document within <Create> that contains the element to be added.

Once an object has been created and loaded into Google Earth, it takes on the URL of the original parent Document of Folder. To perform subsequent updates to objects added with this Update/Create mechanism, set <targetHref> to the URL of the original Document or Folder (not the URL of the file that loaded the intervening updates).

<Delete>

Deletes features from a complex element that has already been loaded via a <NetworkLink>. The <targetHref> element in <Update> specifies the *.kml* or *.kmz* file containing the data to be deleted. Within that file, the element to be deleted must already have an explicit **id** defined for it. The <Delete> element references this **id** in the **targetId** attribute.

Child elements for <Delete>, which are the only elements that can be deleted, are Document, Folder, GroundOverlay, Placemark, and ScreenOverlay.

Example of <Change>

```
<NetworkLinkControl>
  <Update>
    <targetHref>http://www/~sam/January14Data/Point.kml</targetHref>
    <Change>
      <Point targetId="point123">
        <coordinates>-95.48,40.43,0</coordinates>
      </Point>
    </Change>
  </Update>
</NetworkLinkControl>
```

Example of <Create>
This example creates a new Placemark in a previously created Document that has an **id** of "region24." Note that if you want to make subsequent updates to "placemark891," you will still use *http://myserver.com/Point.kml* as the <targetHref>.

```
<Update>
  <targetHref>http://myserver.com/Point.kml</targetHref>
  <Create>
    <Document targetId="region24">
      <Placemark id="placemark891">
        <Point>
          <coordinates>-95.48,40.43,0</coordinates>
        </Point>
      </Placemark>
    </Document>
  </Create>
</Update>
```

Example of <Delete>
This example deletes a Placemark previously loaded into Google Earth. (This Placemark may have been loaded directly by a NetworkLink with the specified URL, or it may have been loaded by a subsequent Update to the original Document.)

```
<Update>
  <targetHref>http://www.foo.com/Point.kml</targetHref>
  <Delete>
    <Placemark targetId="pa3556"></Placemark>
  </Delete>
</Update>
```

Contained By

- <NetworkLinkControl>

<Url>

Note: This element was deprecated in KML Release 2.1 and is replaced by <Link>, which provides the additional functionality of Regions. The <Url> tag will still work in Google Earth, but use of the newer <Link> tag is encouraged.

Use this element to set the location of the link to the KML file, to define the refresh options for the server and viewer changes, and to populate a variable to return useful client information to the server.