Open Geospatial Consortium Inc.

Date: 2007-09-14

Reference number of this OGC® project document: **OGC 07-113r1**

Version: 0.0.14

Category: OGC® Best Practices

Editor: Tim Wilson

KML 2.2 – An OGC Best Practice

Warning

This is an OGC Best Practices Document. It is not an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an official position of the OGC membership on this particular technology topic.

Document type:	OGC® Best Practices Document
Document subtype:	
Document stage:	Draft
Document language:	English

Preamble to "KML 2.2 – An OGC Best Practice"

Google submitted KML (formerly Keyhole Markup Language) to the Open Geospatial Consortium (OGC) to be evolved within the OGC consensus process with the following goal: KML Version 2.2 will be an adopted OGC implementation standard. Future versions may be harmonized with relevant OpenGIS standards that comprise the OGC standards baseline. There are four objectives for this standards work:

- That there be one international standard language for expressing geographic annotation and visualization on existing or future web-based online and mobile maps (2d) and earth browsers (3d).

- That KML be aligned with international best practices and standards, thereby enabling greater uptake and interoperability of earth browser implementations.

- That the OGC and Google will work collaboratively to insure that the KML implementer community is properly engaged in the process and that the KML community is kept informed of progress and issues.

- That the OGC process will be used to insure proper life-cycle management of the KML candidate standard, including such issues as backwards compatibility.

The OGC has developed a broad Standards Baseline. Google and the OGC believe that having KML fit within that family will encourage broader implementation and greater interoperability and sharing of earth browser content and context.

KML is an XML language focused on geographic visualization, including annotation of maps and images. Geographic visualization includes not only the presentation of graphical data on the globe, but also the control of the user's navigation in the sense of where to go and where to look.

From this perspective, KML is complementary to most of the key existing OGC standards including GML (Geography Markup Language), WFS (Web Feature Service) and WMS (Web Map Service). Currently, KML (2.2) utilizes certain geometry elements derived from GML (version 2.1.2). These elements include point, line string, linear ring, and polygon.

The OGC and Google have agreed that there can be additional harmonization of KML with GML (e.g. to use the same geometry representation) in the future. The Mass Market Geo Working Group (MMWG) in the OGC will establish such additional harmonization activities. OGC specifications such as Context and Styled Layer Descriptor (SLD) may be considered.

Google initially submitted the KML 2.1 Reference Manual to the OGC. Carl Reed of OGC reformatted this manual into the OGC Best Practices Document Template. During the April 2007 Technical Committee meetings, the OGC membership approved the KML 2.1 OGC Best Practices Paper (OGC 07-039r1). During the June 2007 Technical Committee meeting the MMWG approved updating this document to KML 2.2 and adding informative text to further describe the KML coordinate reference system (CRS) and geometry models.

This document is based primarily on the current KML 2.2 reference documentation from Google. This OGC Best Practices document is provided to the community to initiate the process of KML standardization within the OGC consensus process.

Contents

i. Preface

This document is being submitted by Google, Inc. to the OGC for future consideration as an OGC® Implementation Standard and supersedes the KML 2.1 version of this document (OGC 07-039r1).

During the June 2007 Technical Committee meeting the MMWG outlined the following process to complete the KML 2.2 OGC Implementation Specification development:

- Update the KML 2.1 Best Practices Paper to KML 2.2, and add informative text describing the KML coordinate reference system (CRS) and geometry models in terms of the GML Geometry and Topic 2 (Spatial Referencing) and coordinate reference systems (CRS). This revision will be submitted for approval at the September 2007 Technical Committee meeting.

- Establish a KML 2.2 Standards Working Group (SWG) at the September 2007 Technical Committee meeting. The mandate of this group will be to prepare and submit a KML 2.2 RFC, based on the KML 2.2 Best Practices Paper, for consideration as an OGC adopted Implementation Standard in early 2008.

ii. Submitting organizations

The following organizations submitted this Best Practices Paper to the Open Geospatial Consortium Inc.:

a) Google, Inc.

b) Galdos Systems Inc.

iii. Submission contact points

All questions regarding this submission should be directed to the editor or submitters:

CONTACT	COMPANY	EMAIL
Tim Wilson	Galdos Systems Inc.	twilson at galdosinc.com
David Burggraf	Galdos Systems Inc.	dburggraf at galdosinc.com
Ron Lake	Galdos Systems Inc.	rlake at galdosinc.com
Susan Patch	Galdos Systems Inc.	spatch at galdosinc.com
Brian McClendon	Google, Inc.	bam at google.com
Michael Jones	Google, Inc.	mtj at google.com
Michael Ashbridge	Google, Inc.	mashbridge at google.com
Bent Hagemark	Google, Inc.	bent at google.com
Josie Wernecke	Google, Inc.	josiew at google.com
Carl Reed	Open Geospatial Consortium	creed at opengeospatial.org

iv. Revision history

Date	Release	Author	Paragraph modified	Description
3-7-06	0.0.5	Carl Reed	New	Initial Version
4/17/07	0.0.9	Carl Reed	Various	Updates based on comments received during Mass Market GEO WG meeting as well as a request from Google to remove 3 elements.
5/2/07	0.0.9	Carl Reed	Various	Add preamble and edit document for posting as a BP.
8/1/07	0.0.10	Susan Patch	Various	Corrected all links to target within the document Modified text to include final Google KML 2.2 reference text.
8/29/07	0.0.11	Tim Wilson	Various	Modified text to describe KML in terms of generic earth browsers. Updated text and terminology to OGC 06-135r1 standards.

Date	Release	Author	Paragraph modified	Description
8/29/07	0.0.12	David Burggraf	Added Section 6, Annex A and various other edits.	Added informative text describing the KML coordinate reference system (CRS) and geometry models in terms of the GML Geometry and Topic 2 (Spatial Referencing). Added GML CRS dictionary to Annex A. Note: Document number changed to 07-113 when submitted to OGC.
9/6/07	0.0.13	David Burggraf, Ron Lake, Tim Wilson, Susan Patch	Various	Modified text for clarity and consistency and to further describe KML in terms of generic earth browsers. New diagrams added to Section 6.
9/10/07	0.0.14	Carl Reed	OGC terminology	Adjusted OGC terminology for new OGC (TBD) policies.
9/11/07	0.0.14	Bent Hagemark, Michael Ashbridge	All	Reviewed for errors and omissions.
9/12/07	0.0.14	David Burggraf	Section 6, geometry elements	Clarification of definitions.
9/12/07	0.0.14	Ron Lake	Camera, PhotoOverlay, Model	Clarification of definitions.
9/14/07	0.0.14	Susan Patch	All	Copy editing and application of OGC styles.
9/14/07	0.0.14	Tim Wilson	All	Corrected errors and improved terminology.

v. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

Introduction

KML is an XML grammar used to visualize geographic data in an earth browser. KML uses a tag-based structure with nested elements and attributes and is based on the XML standard.

The KML community is wide and varied. Casual users create KML Placemarks to identify their homes, describe journeys, and plan cross-country hikes and cycling ventures. Scientists use KML to provide detailed mappings of resources, models, and trends such as volcanic eruptions, weather patterns, earthquake activity, and mineral deposits. Real estate professionals, architects, and city development agencies use KML to propose construction and visualize plans. Students and teachers use KML to explore people, places, and events, both historic and current. Organizations such as National Geographic, UNESCO, and the Smithsonian have all used KML to display their rich sets of global data.

KML documents and their related images (if any) may be compressed using the ZIP format into KMZ archives. KML documents and KMZ archives may be shared by e-mail, hosted locally for sharing within a private internet, or hosted on a web server.

KML 2.2 – An OGC Best Practice

1 Scope

KML is an XML grammar used to visualize geographic data in an earth browser, such as a 3D virtual globe and 2D web browser or mobile mapping applications. A KML instance is processed in much the same way that HTML (and XML) documents are processed by web browsers. Like HTML, KML has a tag-based structure with names and attributes used for specific display purposes.

KML can be used to:

- Annotate the Earth
- Specify icons and labels to identify locations on the surface of the planet
- Create different camera positions to define unique views for KML features
- Define image overlays to attach to the ground or screen
- Define styles to specify KML feature appearance
- Write HTML descriptions of KML features, including hyperlinks and embedded images
- Organize KML features into hierarchies
- Locate and update retrieved KML documents from local or remote network locations
- Define the location and orientation of textured 3D objects

2 Conformance

There are no conformance clauses for this Best Practices Document.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this part of OGC 07113r1. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply; however, parties to agreements based on this part of OGC 07113r1 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

KML 2.2 Reference Document, 2007, Google Inc.
http://code.google.com/apis/kml/documentation/kml_reference.html

KML 2.2 Schema Document, 2007, Google Inc.
http://code.google.com/apis/kml/schema/kml22beta.xsd

Atom Syndication Format: http://atompub.org.

OASIS xAL, Extensible Address Language, http://www.oasis-open.org/committees/ciq/ciq.html#6

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601:2004, Data elements and interchange formats — Information interchange — Representation of dates and times

ISO 19107:2003, Geographic Information — Spatial schema

ISO 19111:—1), Geographic Information — Spatial referencing by coordinates

IETF RFC 1808, Relative Uniform Resource Locators (June 1995)

IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax. (August 1998)

W3C HTML and URLs: http://www.w3.org/TR/WD-html40-970708/htmlweb.html

W3C XML, Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation (4 February 2004)

W3C XML Namespaces, Namespaces in XML. W3C Recommendation (14 January 1999)

W3C XML Schema Part 1, XML Schema Part 1: Structures. W3C Recommendation (2 May 2001)

W3C XML Schema Part 2, XML Schema Part 2: Datatypes. W3C Recommendation (2 May 2001)

4 Terms and symbols

4.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1.1

application schema

conceptual schema for data required by one or more applications.

[ISO 19101]

4.1.2

attribute <XML>

name/value pair contained in an element

NOTE: In this document an attribute is an XML attribute unless otherwise specified

4.1.3

boundary

set that represents the limit of an entity

[ISO 19107]

4.1.4

bounding box

minimum volume that encloses a set of objects or data points.

4.1.5

child element <XML>

immediate descendant element

4.1.6

coordinate

one of a sequence of n numbers designating the position of a point in n-dimensional space

[ISO 19111]

NOTE: In a coordinate reference system, the n numbers shall be qualified by units.

4.1.7

coordinate reference system

coordinate system that is related to an object by a datum

[ISO 19111]

4.1.8

coordinate system

set of mathematical rules for specifying how coordinates are to be assigned to points

[ISO 19111]

4.1.9

coordinate tuple

tuple composed of a sequence of coordinates

[ISO 19111]

4.1.10

data type

specification of a value domain with operations allowed on values in this domain

[ISO/TS 19103]

EXAMPLE: Integer, Real, Boolean, String, Date (conversion of a data into a series of codes).

NOTE: Data types include primitive predefined types and user-definable types. All instances of a data types lack identity.

4.1.11

datum

parameter or set of parameters that define the position of the origin, the scale, and the orientation of a coordinate system

[ISO 19111]

NOTE: A datum may be a geodetic datum, a vertical datum, an engineering datum, an image datum or a temporal datum.

4.1.12

document <XML>

well-formed XML instance

4.1.13

earth browser

software for displaying and annotating models of the Earth

4.1.14

element <XML>

basic information item of an XML document containing child elements, attributes and character data

NOTE: From the XML Information Set: "Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its 'generic identifier' (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value."

4.1.15

field

child element or attribute of simple content

4.1.16

geodetic datum

datum describing the relationship of a 2- or 3-dimensional coordinate system to the Earth

[ISO 19111]

4.1.17

geographic view

display of geographic KML elements

4.1.18

interior

set of all points that are on a geometric object but which are not on its boundary

4.1.19

line string

curve composed of straight-line segments

4.1.20

list view

display of one or more hierarchies of KML Features

4.1.21

namespace <XML>

collection of names, identified by a URI reference, which are used in XML documents as element names and attribute names [W3C XML Namespaces]

4.1.22

plate carrée projection

a plane projection in which the horizontal coordinate is the longitude and the vertical coordinate is the latitude

4.1.23

point

0-dimensional geometric primitive, representing a position

[ISO 19107]

NOTE: The boundary of a point is the empty set.

4.1.24

polygon

planar surface defined by 1 exterior boundary and 0 or more interior boundaries

4.1.25

resource

a local or remote network data object or service that is identified by a URL

4.1.26

schema

formal description of a model

[ISO 19101]

NOTE: In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the relationship between the attributes and elements of an XML object (for example, a document or a portion of a document)

4.1.27

schema <XML Schema>

collection of schema components within the same target namespace

EXAMPLE: Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

4.1.28

schema document <XML Schema>

XML document containing schema component definitions and declarations

NOTE: The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

4.1.29

tag <XML>

markup in an XML document delimiting the content of an element

NOTE: A tag with no forward slash (e.g. <Placemark>) is called a start-tag (also opening tag), and one with a forward slash (e.g. </Placemark> is called an end-tag (also closing tag).

4.1.30

tuple

ordered list of values

4.1.31

Uniform Resource Identifier (URI)

unique identifier for a resource, structured in conformance with IETF RFC 2396

NOTE: The general syntax is <scheme>::<scheme-specific-part>. The hierarchical syntax with a namespace is <scheme>://<authority><path>?<query> – see [RFC 2396].

4.2 Acronyms (and abbreviated terms)

Some frequently used abbreviated terms:

COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
CS	Coordinate System
CSV	Comma Separated Values
CT	Coordinate Transformation
DTD	Document Type Definition

EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language

NOTE: The acronym GML was previously used in ISO also as Generalized Markup Language (which led to SGML Standard Generalized Markup Language, ISO 8879).

HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
RFC	Request for Comments
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
xAL	eXtensible Address Language
XML	eXtended Markup Language
XSD	XML Schema Definition
0D	Zero Dimensional
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

5 Conventions

5.1 UML Notation

There is no UML associated with this candidate specification.

5.2 XML Namespaces

All components of the KML schema are defined in the namespace with the identifier "`http://earth.google.com/kml/2.2`", for which the prefix kml or the default namespace is used within this Best Practice Document.

NOTE: When KML becomes an adopted OGC® Implementation Specification the namespace is expected to change to "`http://www.opengis.net/kml/2.2`".

5.3 Versioning

Each schema document specifying components of the KML schema shall carry a version attribute as defined in the XML Schema Recommendation. The format of the version attribute string is x.y.z where x denotes the major version number, y denotes a minor version number, and z denotes a bug fix release for that document. All versions with the same major version are compatible. For this reason, all KML 2.2 files validate using the KML 2.2 schema.

The KML version described by this Best Practices Paper is 2.2.

5.4 Deprecated parts of previous versions of KML

The verb "deprecate" provides notice that the referenced portion of this Best Practices Paper is being retained for backwards compatibility with earlier versions but may be removed from a future version without further notice.

5.5 XML Schema

KML uses the W3C XML Schema language to describe the grammar of conformant KML data instances. Conformant KML 2.2 files shall validate against the KML 2.2 schema.

6 Overview of Spatial Extents and Coordinate Reference Systems

Each element that extends the *<Geometry>* element defines a spatial extent of a `Placemark`. The spatial extent may include the location of an anchor point on the earth to serve as an origin for a 3D object as in the case of the `Model` element or may include the encoding of explicit coordinate tuples in the `coordinates` element in the case of the `Point`, `LineString`, and `LinearRing` elements.

The KML encoding of every location and coordinate tuple uses geodetic longitude, geodetic latitude, and altitude as defined in Annex A by the GML Coordinate Reference System (CRS) with identifier LonLat84_5773. Coordinate tuples that are encoded in the `coordinates` element are called control points. The vertical datum is the WGS84 EGM96 Geoid.

The geometric points that represent a `Placemark` which are not explicitly encoded are called interpolated points. The following subclauses describe the interpolation schemes for the `LineString`, `LinearRing`, and `Polygon` elements.

6.1 LineString and LinearRing Interpolation

The type of interpolation used for the `LineString` and `LinearRing` elements depend on the values of the child `altitudeMode` and `tessellate` elements. If the `altitudeMode` value is not **clampToGround** then the interpolation between two consecutive control points is a straight line segment in the 3D WGS 84 geocentric coordinate reference system (urn:x-ogc:def:crs:EPSG:6.12:4978). This straight line segment will be referred to as *L* in Table 1, which summarizes the `LineString` and `LinearRing` interpolation scheme for the various combinations of `altitudeMode` and `tessellate` values.

Table 1: Interpolation scheme for LineString and LinearRing

<altitudeMode>	*<tessellate>*	*Interpolation between control points*
relativeToGround or absolute	0 (false) or 1 (true)	A straight line segment *L* in the 3D WGS 84 geocentric coordinate reference system (urn:x-ogc:def:crs:EPSG:6.12:4978)
clampToGround	1 (true)	Project each point of *L* to the terrain surface along a line through the earth's center of mass
clampToGround	0 (false)	First project each control point to the terrain surface along a line through the earth's center of mass, then interpolate between the projected control points along a straight line segment in the 3D WGS 84 geocentric coordinate reference system (urn:x-ogc:def:crs:EPSG:6.12:4978)

For example, in the following KML instance the `LineString` coordinates element has two control points (–135,30,500000) and (–80,30,500000) of the form (long,lat,altitude) in the CRS defined in Annex A with gml:id="LonLat84_5773".

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <LineString>
        <altitudeMode>absolute</altitudeMode>
        <coordinates>-135,30,500000 -80,30,500000</coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

Since the `altitudeMode` is **absolute**, the interpolation between these two control points is the straight line segment *L* shown in Figure 1 in the 3D WGS 84 geocentric CRS, which does not follow the earth's curvature and cuts through the earth's terrain.

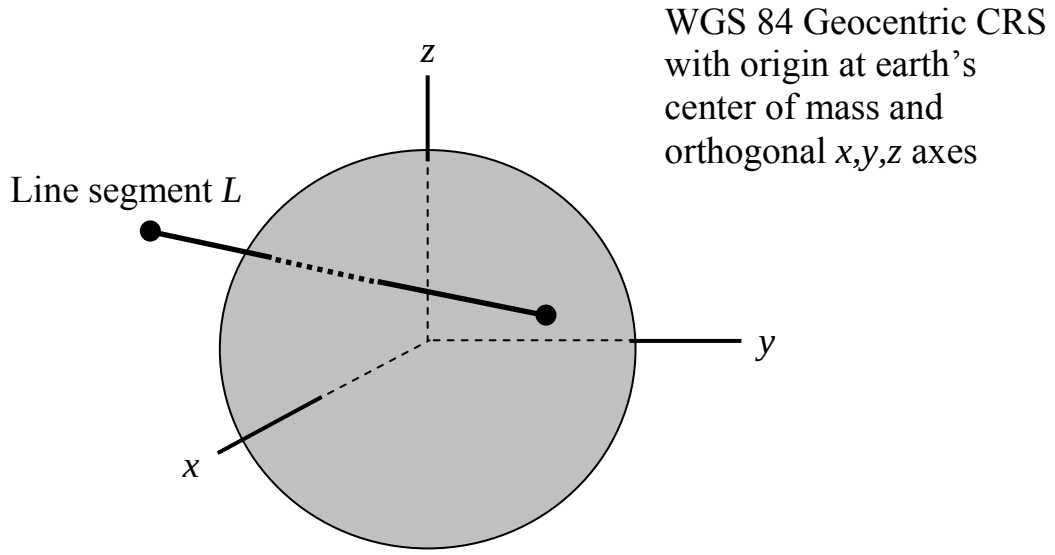


Figure 1: A LineString Comprised of Two Control Points and a Single Line Segment Interpolated in the WGS 84 Geocentric CRS

The line segment *L* will be projected to the terrain surface if `altitudeMode` and `tessellate` are set as in the following `LineString` instance. In this case the projected `LineString` will follow the earth's curvature as shown in Figure 2

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <LineString>
        <tessellate>true</tessellate>
        <altitudeMode>clampedToGround</altitudeMode>
        <coordinates>-135,30,500000 -80,30,500000</coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

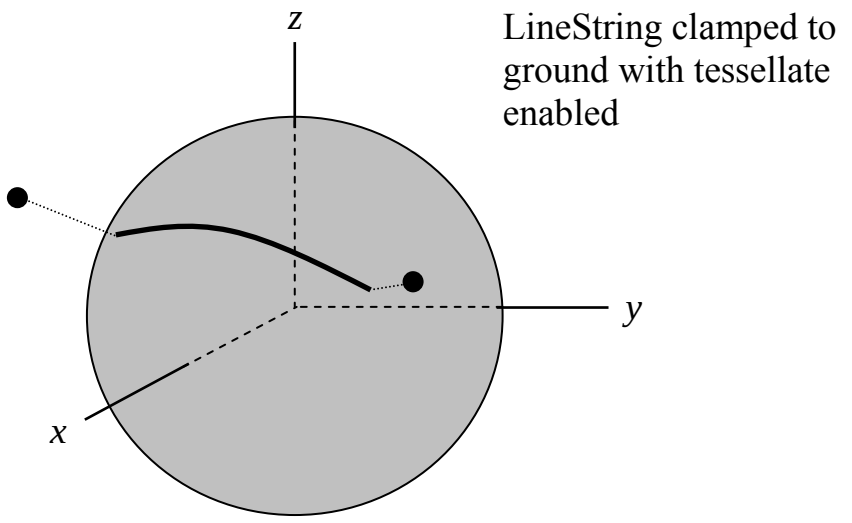


Figure 2: A Line Segment Interpolated in the WGS 84 Geocentric CRS

If `altitudeMode` and `tessellate` are set as in the following `LineString` instance, then the only the control points are projected to the terrain and the interpolalation between the projected control points is a straight line segment in the WGS 84 Geocentric CRS as shown in Figure 3.

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <LineString>
        <tessellate>false</tessellate>
        <altitudeMode>clampedToGround</altitudeMode>
        <coordinates>-135,30,500000 -80,30,500000</coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
<LineString>
```

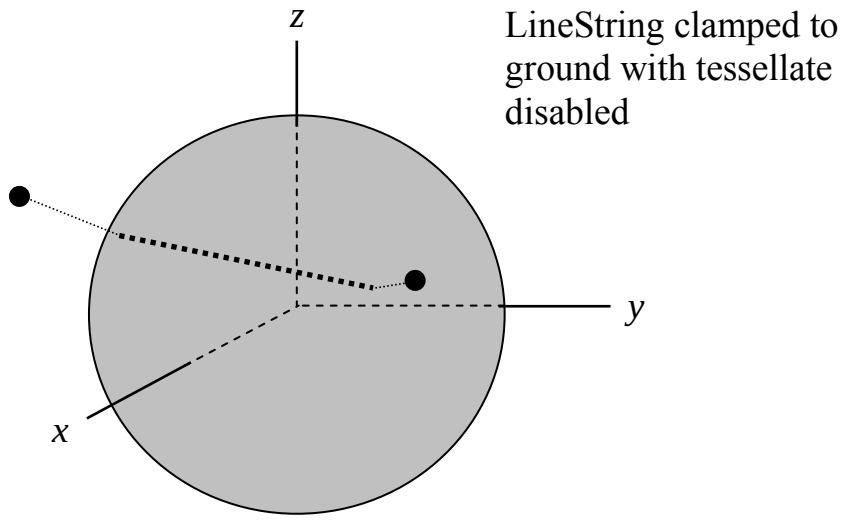


Figure 3: A Line Segment Interpolated in the WGS 84 Geocentric CRS

6.2 Polygon Interpolation

The type of interpolation used for the `Polygon` element also depends on the values of the child `altitudeMode` and `tessellate` elements. If the `altitudeMode` value is not **clampToGround** then the interpolation of the `Polygon` boundary comprised of the descendent `LinearRing` elements is as described previously in Table 1. The remaining interior points of the `Polygon` are then filled in linearly in the 3D WGS 84 geocentric CRS, i.e. they must lie on the plane that passes through all the control points of each `LinearRing`.

NOTE: The control points of every `LinearRing` must lie on a common plane.

Table 2 summarizes the `Polygon` interpolation scheme for the various combination of `altitudeMode` and `tessellate` values.

Table 2: Interpolation scheme for Polygon

<altitudeMode>	*<tessellate>*	*Interpolation between control points*
relativeToGround or absolute	0 (false) or 1 (true)	Boundary points of the Polygon in the descendent LinearRing(s) are interpolated as in Table 1 and the interior points are filled in linearly in the 3D WGS 84 geocentric coordinate reference system (urn:x-ogc:def:crs:EPSG:6.12:4978), i.e. they must lie on a plane.
clampToGround	0 (false)	The boundary control points of each descendent LinearRing are first projected to the plate carrée plane (where altitude is dropped), then straight line segment interpolation in the plate carrée (long,lat) plane is used between consecutive control points. The interior points are then filled in linearly in the plate carrée plane. Finally, the (long,lat) points of the polygon in the plate carrée plane are mapped back to (long.lat,alt) points on the earth's terrain surface model

For example, the following KML `Polygon` encodes five control points in its outer boundary in the CRS defined in Annex A with gml:id="LonLat84_5773".

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <Polygon>
        <altitudeMode>absolute</altitudeMode>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>
-135,50,300000 -135,40,450000 -80,40,450000 -80,50,300000 -135,50,300000
            </coordinates>
          </LinearRing>
        </outerBoundaryIs>
      </Polygon>
    </Placemark>
  </Document>
</kml>
```

Since the `altitudeMode` is **absolute**, the outer boundary points of the polygon that are interpolated between the control points in the `LinearRing` form a quadrilateral perimeter in the 3D WGS 84 geocentric CRS. The interior points of this 4 sided polygon are filled in linearly in the 3D WGS 84 geocentric CRS and form the plane region inside the perimeter. Note that the plane region does not follow the earth's curvature and cuts through the surface of the earth as shown in Figure 4.

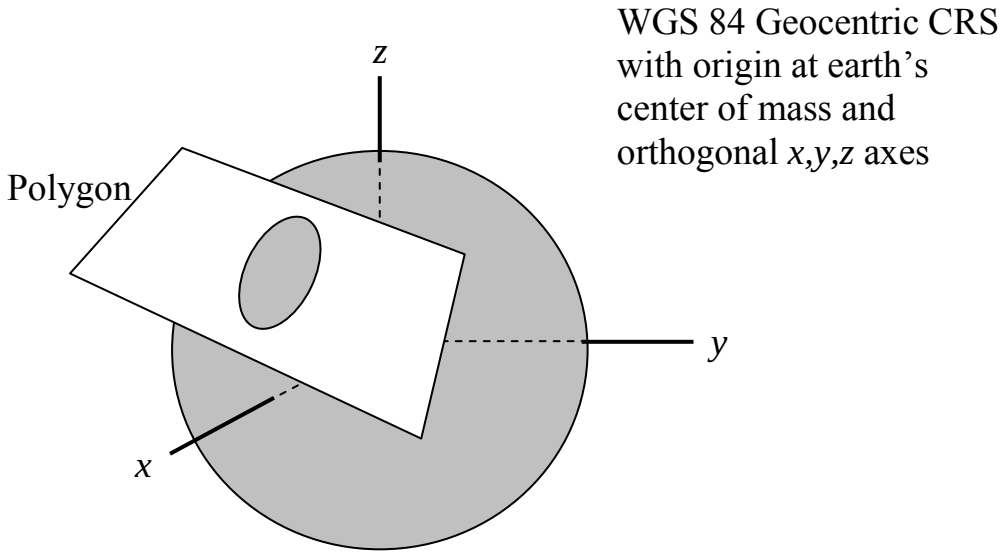


Figure 4: KML Polygon interpolation in the WGS 84 Geocentric CRS

In the following `Polygon` instance the `altitudeMode` is set to **clampToGround**.

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <Polygon>
        <altitudeMode>clampToGround</altitudeMode>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>-135,78.5,300000 -135,12.5,300000 -45,12.5,300000 -
45,78.5,300000 -135,78.5,300000</coordinates>
          </LinearRing>
        </outerBoundaryIs>
      </Polygon>
    </Placemark>
  </Document>
</kml>
```

The outer boundary points of the polygon that are interpolated between the control points in the `LinearRing` form a rectangular perimeter in the plate carrée plane. Then the interior points of the rectangle are filled in linearly in this plane. Finally each (long,lat) point of the rectangle is mapped to a (long,lat,alt) point on the earth terrain surface as shown in Figure 5.

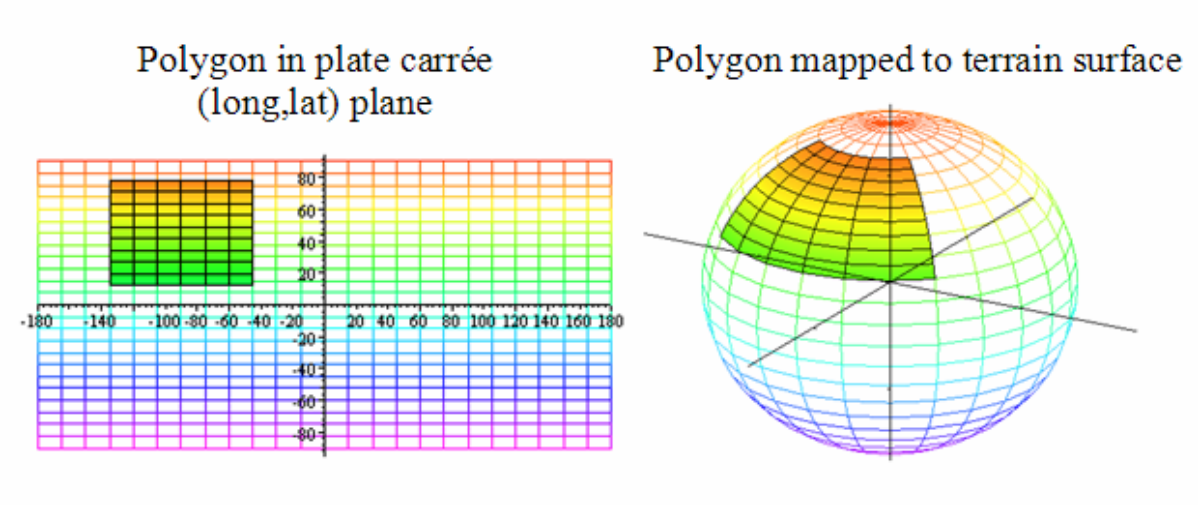


Figure 5: KML Polygon Interpolation in the Plate Carrée Plane

7 Overview of KML Model

The following section provides an overview of the KML model and describes conventions used in this document.

7.1 Document Conventions

Section 8 describes the content model for each KML element that is defined by its normative type definition in the KML schema document. The content model is described in terms of child elements and attributes. Where a child element is described in a separate sub clause a link is provided to it.

Each sub clause of Section 8 includes a syntax section that provides an informative listing of the content model for the described element. This listing uses the following conventions:

- default values for each field. Unless otherwise stated, default values for optional field elements shall apply when such elements are empty or are absent.
- field types. See KML Fields section.

Throughout this document, the following conventions are used:

- Concrete element and attribute names are formatted in `Courier New`, and in *`Courier New Italic`* for abstract elements.
- A value from an enumerated set is in **bold**.
- Child element content is described within the element sub clause whose type declares such children.
- An element is said to extend a second element if the type of the first element derives from the type of the second element in the sense of XML Schema. This terminology is used in the subclauses with the headings Extends and Extended By.
- An element is said to contain a second element if the second is a child element of the first element. This terminology is used in the subclauses with the headings Contains and Contained By for those elements that are defined in their own sub clause.
- The *`Feature`* element refers to the following elements that extend it: `Document`, `Folder`, `GroundOverlay`, `NetworkLink`, `PhotoOverlay`, `Placemark`, and `ScreenOverlay`.

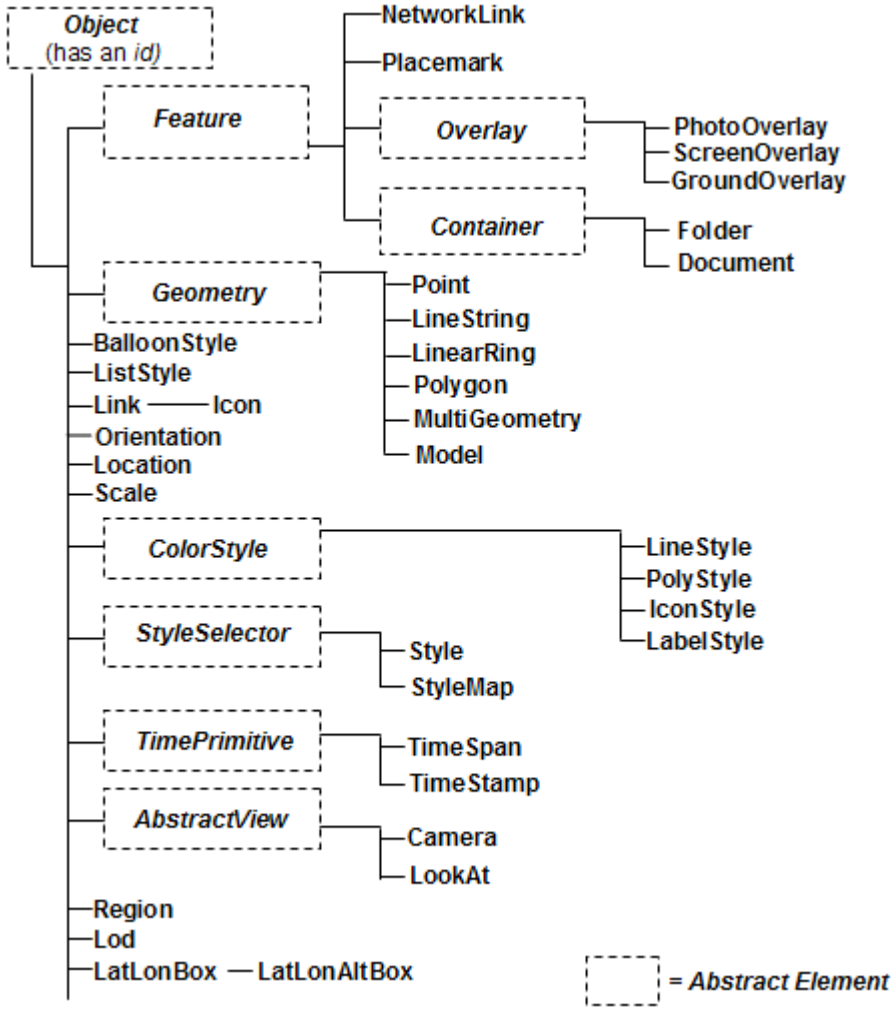
- The *Geometry* element refers to the following elements that extend it: Point, Polygon, LinearRing, LineString, Model, and MultiGeometry.
- The *Object* element refers to the following elements that extend it: Region, Lod, Icon, Link, Location, Orientation, Scale, ResourceMap, LatLonBox, IconStyle, LabelStyle, LineStyle, PolyStyle, BalloonStyle, and ListStyle.
- The *Overlay* element refers to the following elements that extend it: GroundOverlay, PhotoOverlay, and ScreenOverlay.
- The *StyleSelector* element refers to the following elements that extend it: Style and StyleMap.
- The term Document refers to the KML Document element; the term document refers to an XML document.

7.2 Overview of the KML Content Model

This section provides an overview of the KML schema type hierarchy. A diagram illustrating the KML schema types and their relationships is shown below. While element names are used, they represent the schema types that define their content.

In KML, some types are derived from a parent type. A derived type inherits all of the elements of its parent type and may add some specific element content of its own. This is shown in the diagram as follows: 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 adds some elements that are specific to the Placemark element.

7.2.1 KML Schema Hierarchy



KML includes abstract elements whose type is also abstract. Such abstract types are used to establish schema type hierarchies. Abstract elements may serve as placeholders for elements that substitute for them in the XML Schema sense. In the diagram, abstract elements (representing their type) are shown in italics surrounded by dotted boxes.

All concrete elements derived from `Object` may have an assigned identifier.

Because KML is an XML grammar, element names are case-sensitive and must appear exactly as specified in the KML 2.2 schema.

7.2.2 KML Fields

KML fields use common XSD types such as *boolean*, *string*, *double*, *float*, and *int*, and types of simple content defined in the KML schema. The following table lists the KML field types and links to sample elements that use them:

Table 3: KML Field Types

Field Type	*Value*	*Example Use*
CoordinatesType	XSD List of: string	See <coordinates> in <LineString>, <LinearRing>, and <Point>
altitudeModeEnum	Base XSD Type: string *value* comes from list: {'clampToGround'|'relativeToGround'|'absolute'}	See <LookAt> and <Region>
angle180	Base XSD Type: double -180 <= *value* <= 180	See <longitude> in <Model>
angle360	Base XSD Type: double -360 <= *value* <= 360	See <heading>, <tilt>, and <roll> in <Orientation>
angle90	Base XSD Type: double -90 <= *value* <= 90.0	See <latitude> in <Model>
anglepos180	Base XSD Type: double 0 <= *value* <= 180	See <Camera>
anglepos90	Base XSD Type: double 0.0 <= *value* <= 90.0	See <tilt> in <LookAt>
color	Base XSD Type: hexBinary length = 4	See any element that extends <ColorStyle>
colorModeEnum	Base XSD Type: string *value* comes from list: {'normal'|'random'}	See any element that extends <ColorStyle>
dateTimeType	Union of following XSD types: dateTime, date, gYearMonth, gYear	See <TimeSpan> and <TimeStamp>
displayModeEnum	Base XSD Type: string *value* comes from list: {'default'|'hide'}	See <BalloonStyle>

Field Type	*Value*	*Example Use*
gridOriginEnum	Base XSD Type: string *value* comes from list: {'lowerLeft'|'upperLeft'}	See <PhotoOverlay>
itemIconStateEnum	Base XSD Type: string *value* comes from list: {'open'|'closed'|'error'|'fetching0'| 'fetching1'|'fetching2'}	See <ListStyle>
itemIconStateType	List of: kml:itemIconStateEnum	See <ListStyle>
listItemTypeEnum	Base XSD Type: string *value* comes from list: {'radioFolder'|'check'|'checkHideChildren'| 'checkOffOnly'}	See <ListStyle>
refreshModeEnum	Base XSD Type: string *value* comes from list: {'onChange'|'onInterval'|'onExpire'}	See <Link>
shapeEnum	Base XSD Type: string *value* comes from list: {'rectangle'|'cylinder'|'sphere'}	See <PhotoOverlay>
styleStateEnum	Base XSD Type: string *value* comes from list: {'normal'|'highlight'}	See <StyleMap>
unitsEnum	Base XSD Type: string *value* comes from list: {'fraction'|'pixels'|'insetPixels'}	See <hotSpot> in <IconStyle>, <ScreenOverlay>
viewRefreshModeEnum	Base XSD Type: string value comes from list: {'never'|'onRequest'|'onStop'|'onRegion'}	See <Link>
vec2Type	XML attributes: x=double y=double xunits= kml:unitsEnum yunits= kml:unitsEnum	See <hotSpot> in <IconStyle>, <ScreenOverlay>

7.3 Shared Styles

A <*StyleSelector*> contained by a *Feature* is an "inline style" and shall apply only to the *Feature* that contains it. A *StyleSelector* encoded as the child of a Document is a "shared style." A shared style shall have an id value. A shared style applies to any *Feature* that references the style from its child styleUrl element.

If a *Feature* is associated with both an inline and shared style, the inline style shall take precedence.

Shared styles shall only be encoded within a Document. Shared styles are not inherited by any child *Features* of a Document.

For a *StyleSelector* that applies to a Document, the Document itself must explicitly reference a shared *StyleSelector*. For example:

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

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

The following example illustrates the use of a shared style.

```
<kml xmlns="http://earth.google.com/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>
```

7.4 Entity Replacement

Entity substitution is employed as a template mechanism within the `BalloonStyle text` element. Individual values shall be substituted for each instance of the entity, or a null string if no value exists. The source of values for entity substitution is local to the *`Feature`* being styled and any `Schema` elements associated with it. Entity syntax for identifying substitution values is as follows:

> **$[element_or_attribute_name]**, where "element_or_attribute_name" is the name of a field element or attribute of the *`Feature`*. This identifies the value of the field element or attribute.
>
> or **$[name_attribute_of_Data_element]**, where "name_attribute_of_Data_element" is the value of the `name` attribute of a descendant `Data` element of the *`Feature`*. This identifies the value of the child `value` element of the `Data` element.
>
> or **$[name_attribute_of_Data_element/displayName]**, where "name_attribute_of_Data_element" is the value of the `name` attribute of a descendant `Data` element of the *`Feature`*; "/" is a separator; and "displayName" is the value of the `displayName` attribute of the `Data` element. This identifies the value of the `displayName` attribute.
>
> or **$[TYPENAME/TYPEFIELD]**, where "TYPENAME" is the value of the `name` attribute of a descendant `Schema` element of the *`Feature`*; "/" is a separator; and "TYPEFIELD" is the value of the `name` attribute of a child `SimpleField` element of the `Schema` element . This identifies the value of a descendant `SimpleData` element of the *`Feature`* that references the `SimpleField` element.
>
> or **$[TYPENAME/TYPEFIELD/displayName]**, where "TYPENAME" is the value of the `name` attribute of a descendant `Schema` element of the *`Feature`*; "/" is a separator; and "TYPEFIELD" is the value of the `name` attribute of a child `SimpleField` element of the `Schema` element; and "displayName" is the value of the child `displayName` element of the `SimpleField` element. This identifies the value of the `displayName` element.

For example, the **$[name]** and **$[description]** entities in the following `BalloonStyle text` element shall be replaced by the `name` and `description` values of *`Features`* associated with the `BalloonStyle`:

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

For further examples of entity substitution see:
http://code.google.com/apis/kml/documentation/extendeddata.html#entityreplacement.

8 KML Element Descriptions

8.1 *<AbstractView>*

8.1.1 Syntax

```
<!-- abstract element; do not create -->
<!-- AbstractView id="ID" targetId="NCName" -->          <!-- Camera, LookAt -->
  <!-- extends Object -->
<-- /AbstractView -->
```

8.1.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause. An earth browser displays KML graphics overlaid on a backdrop image which is typically that of the earth. In addition to describing the overlaid graphic elements, KML can define a geographic view of the overlaid graphics and the backdrop image. KML provides two ways to define this view, namely `LookAt` and `Camera`. Both of the elements are derived from *`AbstractView`*.

8.1.3 Extends

- *<Object>*

8.1.4 Extended By

- <Camera>
- <LookAt>

8.1.5 Contained By

- *<Feature>*
- <NetworkLinkControl>

8.2 <BalloonStyle>

8.2.1 Syntax

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

8.2.2 Description

Specifies how the description balloon for a `Feature` is drawn.

8.2.3 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, to apply a blue color with 50 percent opacity to an overlay, specify the following: <bgColor>7fff0000</bgColor>, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00.

NOTE: `bgColor` should be used instead of the deprecated `color` element that it replaces.

<textColor>

Foreground color of the text.

<text>

Text displayed in the balloon.

Text shall support entity substitution as defined in Entity Replacement.

<displayMode>

If `displayMode` is **default**, the balloon shall be displayed. If `displayMode` is **hide**, the

balloon shall not be displayed.

8.2.4 Examples

```
<kml xmlns="http://earth.google.com/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>
```

8.2.5 Extends

- *<Object>*

8.2.6 Contained By

- <Style>

8.3 <Camera>

8.3.1 Syntax

```
<Camera id="ID" targetID="NCName">
  <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 -->
</Camera>
```

8.3.2 Description

The `Camera` element specifies the position and orientation of a virtual camera. This can be used to specify views of the earth or of objects in space.

8.3.2.1 Defining a View

The `Camera` element specifies the position of the view point of the camera using the child elements `longitude`, `latitude`, `altitude` and `altitudeMode`. The orientation of the camera is specified using the additional child elements `heading`, `tilt` and `roll`.

The initial or zero position of the camera is defined by an earth-fixed frame lying in a meridian plane (plane containing the view point, the poles, and the earth's center of mass), with the Z'-axis normal to the earth's surface, the Y'-axis directed away from the equator, and the X'-axis such as to form a right handed orthogonal frame.

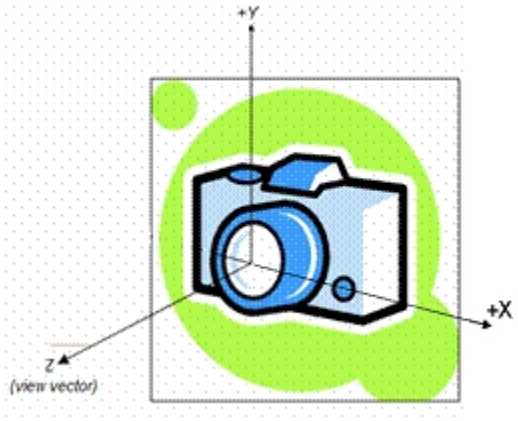
A body fixed frame is assumed attached to the virtual camera, with the Z-axis along the optical axis of the camera, the Y-axis through the top of the camera, and the X-axis such as to form a right handed orthogonal frame. In the zero position of the camera, the camera body Z-axis is aligned with the –Z' axis and the body Y axis with the Y' axis. The orientation of the camera is then defined by the following sequence of rotations (Euler angles) which must be performed in the stated order:

- <altitude> – translate along the Z' axis to the specified altitude.
- <heading> – clockwise rotation around the Z axis. The range of the heading is from 0 to 360 degrees.

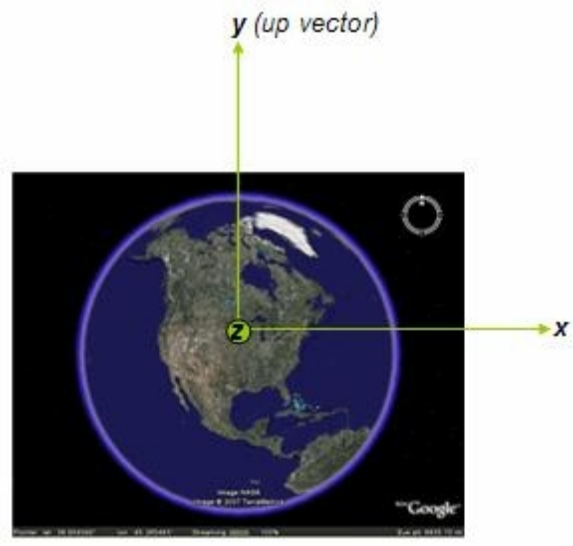
- <tilt> – counter clockwise rotation around the X axis. The range of the tilt is from -180 to + 180 degrees.

- <roll> – clockwise around the Z axis (again). The range of the roll is from -180 to +180 degrees

The camera body axes are shown as follows:



The earth-fixed frame specifying the initial (zero) orientation of the camera is illustrated below:



8.3.2.2 Order of Rotation

The order of rotation is important. By default, the camera shall look straight down the Z axis towards the Earth. The order of rotation is:

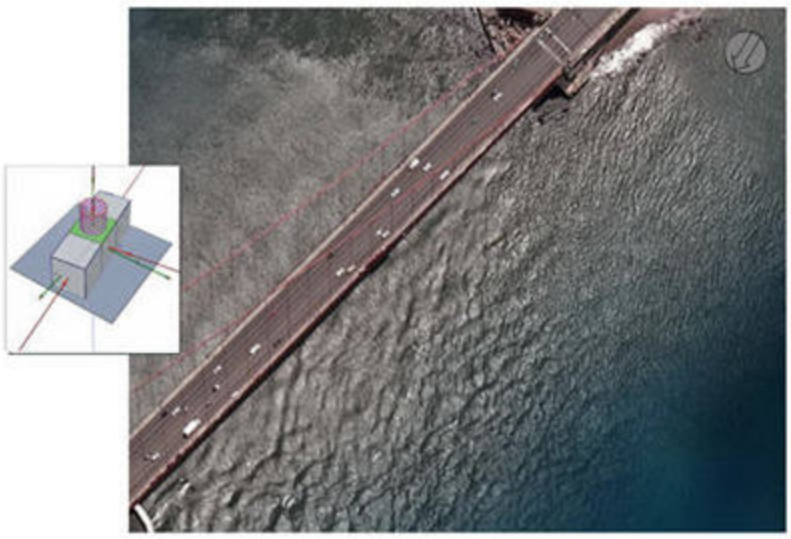
1. **<heading>** – rotate around the Z axis.

2. **<tilt> –** rotate around the X axis.

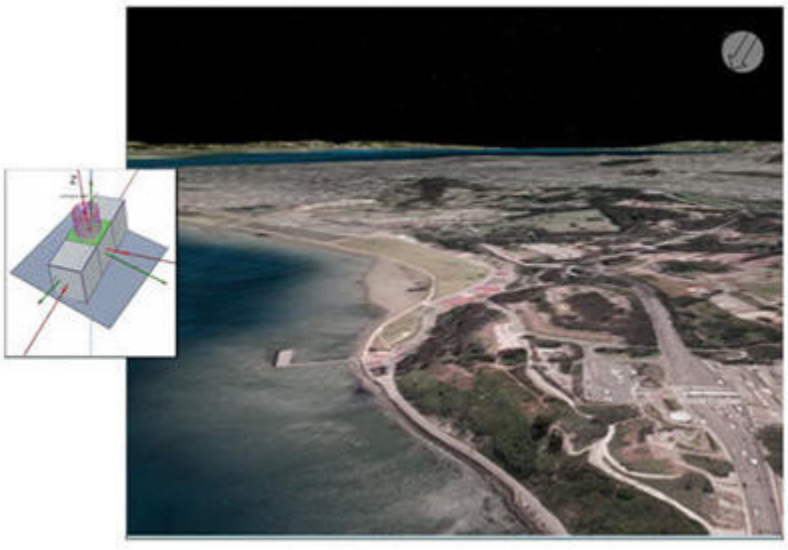
3. **<roll>** – rotate around the Z axis (again).

The camera's view direction is a vector that is computed from these three rotations. Note that each time a rotation is applied, two of the camera axes shall change their orientation.

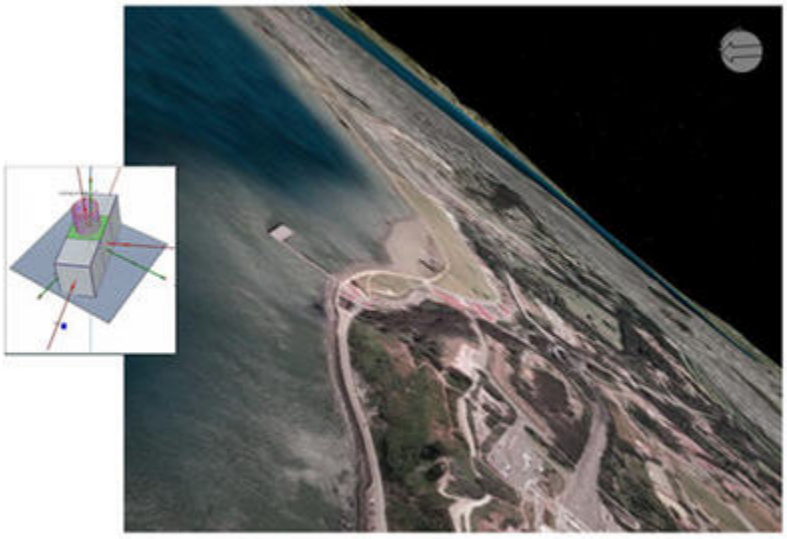
Heading (Rotation about Z):



Tilt (Rotation about X):



Roll (Rotation about Z again)



8.3.3 Elements Specific to Camera

<longitude>

Longitude of the virtual camera (eye point). Angular distance in decimal 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. Decimal 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. See `altitudeMode` for how this value is interpreted.

<heading>

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

<tilt>

Rotation, in decimal 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 of 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 decimal 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** – Interprets the `altitude` as a value in meters above the terrain.
- **clampToGround** – Indicates to ignore the `altitude` specification and place the `Camera` position on the terrain.
- **absolute** – Interprets the `altitude` as a value in meters above the vertical datum.

8.3.4 Extends

- *<AbstractView>*

8.3.5 Contained By

- <Feature>
- <NetworkLinkControl>

8.4 <*ColorStyle*>

8.4.1 Syntax

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

8.4.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause. It provides elements for specifying the color and color mode of style types that derive from it.

8.4.3 Elements specific to ColorStyle

<color>

Color of the graphic element. 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, to apply a blue color with 50 percent opacity to an overlay, specify the following: <color>7fff0000</color>, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00.

<colorMode>

Specifies color mode of the graphic element. 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 transparent white (00ffffff).
- If a single color component is specified (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 values for two or for all three color components are specified, 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.

8.4.4 Extends

- *<Object>*

8.4.5 Extended By

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

8.5 <*Container*>

8.5.1 Syntax

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

<!-- /Container -->
```

8.5.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause. A *Container* holds zero or more *Features* and allows the creation of one or more nested hierarchies of KML features.

For convenience in constructing KML feature hierarchies, the value of the following *Feature* elements shall be inherited by all *Feature* members of a hierarchy: atom:author, atom:link, Region, and *TimePrimitive*, unless overruled by the presence of such elements locally. Thus it is not necessary for a child *Feature* to carry any of these elements where their local value is the same as that of its parent *Feature*. Inheritance of these elements continues to any depth of nesting, but if overruled by a local declaration, then the new value is inherited by all its children in turn. Notwithstanding this rule, such elements may be used locally even if they have the same value as that of a parent *Feature*.

8.5.3 Extends

- *<Feature>*

8.5.4 Extended By

- <Document>
- <Folder>

8.6 <Document>

8.6.1 Syntax

```
<Document id="ID" targetId="NCName">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

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

8.6.2 Description

A `Document` is a container for KML features, shared styles, and user-defined schemas. See also Shared Styles.

8.6.3 Elements Specific to Document

<Schema>

See <Schema>.

<Feature>

See *<Feature>*.

8.6.4 Extends

- *<Container>*

8.6.5 Contains

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

8.7 <ExtendedData>

8.7.1 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>
```

8.7.2 Description

The `ExtendedData` element offers three mechanisms for adding user-defined data to a KML *`Feature`*. These mechanisms are

- Adding untyped name/value data pairs using the `Data` element
- Adding instances of typed fields defined in the user-defined <Schema> element
- Including any XML content defined in namespaces other than the KML namespace and null namespace

These mechanisms can be used concurrently within a single *`Feature`* or KML document.

8.7.3 Elements Specific to ExtendedData

ExtendedData child elements support entity substitution. See Entity Replacement.

<Data name ="string">

Creates an untyped name/value data pair, where:

- The data pair is identified by the `name` attribute
- The value of the data pair is supplied by `value`

<displayName>

An alternate display name for the data pair.

<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">

Encodes an instance of a user-defined data type defined by a referenced `Schema`.

A `SchemaData` element shall reference a `Schema` element using the `schemaUrl` attribute. The value of `schemaURL` should be a full URL, a reference to a `Schema` **id** attribute 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 scope of `ExtendedData` is restricted to its parent *`Feature`* only.

<SimpleData name="string">

Encodes an instance of a user-defined field defined by a referenced `SimpleField`.

The `SimpleData` `name` attribute shall be used to reference the `SimpleField` by name. The identified `SimpleField` shall be declared within the `Schema` element that is referenced from the `SchemaURL` attribute.

The value of `SimpleData` shall be of the data type defined by the referenced `SimpleField`.

Here is an example of encoding two user-defined 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>-123.095524,49.368072,0</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>-123.098414.371120,0</coordinates>
  </Point>
</Placemark>
```

8.7.4 Other XML Content

`ExtendedData` may include any other well-formed, namespace-qualified XML content that is not from:

- the KML namespace, "`http://earth.google.com/kml/2.2`"
- the null or empty namespace, ""

The following example demonstrates the encoding of XML content from the "`http://www.acme.com/app`" namespace:

```
<kml
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://earth.google.com/kml/2.2
                        http://code.google.com/apis/kml/schema/kml22beta.xsd
                      http://www.acme.com/app
                        http://www.acme.com/app/schema/app.xsd"
  xmlns="http://earth.google.com/kml/2.2"
  xmlns:app="http://www.acme.com/app"
  xmlns:gml="http://www.opengis.net/gml">
  <Placemark>
    <name>A road</name>
    <ExtendedData>
      <app:Road>
        <app:numberOfLanes>2</app:numberOfLanes>
        <app:pavement>gravel</app:pavement>
      </app:Road>
    </ExtendedData>
  </Placemark>
</kml>
```

8.7.5 Contained By

- *<Feature>*

8.7.6 See Also

- <Schema>

8.8 <*Feature*>

8.8.1 Syntax

```
<!-- abstract element; do not create -->
<!-- Feature id="ID" targetId="NCName" -->  <!-- Document,Folder,
                                                NetworkLink,Placemark,
                                                GroundOverlay,PhotoOverlay,
                                                ScreenOverlay -->
  <name>...</name>                        <!-- string -->
  <visibility>1</visibility>              <!-- boolean -->
  <open>0</open>                          <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                  <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>               <!-- string -->
  <phoneNumber>...</phoneNumber>          <!-- string -->
  <Snippet maxLines="2">...</Snippet>     <!-- string -->
  <description>...</description>          <!-- string -->
  <AbstractView>...</AbstractView>        <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>      <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>                <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

<-- /Feature -->
```

8.8.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause.

8.8.3 Elements specific to Feature

<name>

Label for the *Feature*.

<visibility>

Boolean value. Specifies whether the *Feature* shall be drawn in the geographic view when it is initially loaded (1), or not (0). In order for a *Feature* to be visible, the <visibility> tag of all its ancestors shall also be set to 1.

<open>

Boolean value. Specifies whether a `Folder` appears closed or open when first loaded into the list view. 0=collapsed, 1=expanded. This element applies only to `Document` and `Folder`. See also <ListStyle>.

<atom:author>

Specifies the author of the *`Feature`*.

This element is defined in the Atom Syndication Format. When used in a KML instance this element shall be associated with the Atom namespace `xmlns:atom="http://www.w3.org/2005/Atom".`

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

Specifies the URL of the web resource (KML or KMZ resource) that contains the *`Feature`*. The URL is encoded as the value of the `href` attribute.

This element is defined in the Atom Syndication Format. When used in a KML instance this element shall be associated with the Atom namespace `xmlns:atom="http://www.w3.org/2005/Atom".`

<address>

A string value representing an unstructured address for the *`Feature`* such as street, city, state address, and/or a postal code. This may be used to geocode the location of a *`Feature`* if it does not contain a *`Geometry`* element.

<xal:AddressDetails>

A structured address for the *`Feature`* formatted according to xAL, or eXtensible Address Language, an international standard for address formatting. This may be used to geocode the location of a *`Feature`* if it does not contain a *`Geometry`* element.

When used in a KML instance this element shall be associated with the xAL namespace `"urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"`

<phoneNumber>

A string value representing a telephone number. The number should be formatted according to RFC2806.

<Snippet maxLines="2" >

A short description of the *`Feature`*. This may be used instead of `description` in the list view.

The `maxLines` attribute specifies the maximum number of lines to display in the list view.

<description>

A description of the *`Feature`*. This should be displayed in the description balloon.

If `description` includes the HTML **<a href="…" type="…">** tag, it should be interpreted as follows:

- The `href` attribute specifies a URL.
- If the target of the `href` is a KML resource, an earth browser should load the resource if the link is activated.

The `href` may reference another *`Feature`* if its value follows the fragment URL convention; that is, a URL with a # sign followed by a KML identifier. See W3C "HTML and URLs". If such a link is activated the geographic view should fly to the *`Feature`* whose ID matches the fragment. If this Feature has a `LookAt` or `Camera` element, it shall be viewed from the specified viewpoint.

The display of the target *`Feature`* 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 resource *CraftsFairs.kml* resource, fly to the `Placemark` whose ID is "Albuquerque," and open its balloon:

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

The `type` attribute specifies the MIME type for the target resource. An earth browser should interpret the target resource according to this specified MIME type when attempting to load it. To indicate that the target resource is KML specify the following MIME type:

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

To indicate that the target resource is a KMZ archive specify the following MIME type:

```
type="application/vnd.google-earth.kmz"
```

For example, the `type` attribute below indicates that an earth browser should attempt to load the target as a KML resource 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*>

Specifies a viewpoint for the *`Feature`*. See <*AbstractView*>.

<*TimePrimitive*>

Specifies an effective time stamp or period for the *`Feature`*. See <*TimePrimitive*>.

<styleUrl>

Reference to a `Style` or `StyleMap`. The reference shall be encoded as a fragment URL. The value of the fragment ID shall be the **id** 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.

A fragment URL would appear as:

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

A full URL with # referencing would appear as:

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

See also <Style>, <StyleMap>, and Shared Styles.

<*StyleSelector*>

One or more `Styles` or `StyleMaps` used to style the *`Feature`*. See <*StyleSelector*> and Shared Styles.

<Region>

Affects the visibility of the *`Feature`*. A *`Feature`* associated with a `Region` is drawn only when the `Region` is active. See <Region>.

<ExtendedData>

Allows for the addition of user-defined data to a KML file. See <ExtendedData>.

NOTE: `ExtendedData` should be used instead of the deprecated `Metadata` element that it replaces.

8.8.4 Sample Use of HTML Elements within a Description

```
<kml xmlns="http://earth.google.com/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>
```

8.8.5 Sample Use of Atom Elements

This example shows use of the `atom:author`, `atom:name` and `atom:link` attribution elements from the Atom namespace. In this case, `atom:author` and `atom:link` apply to both `Placemarks`.

```
<kml xmlns="http://earth.google.com/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>
```

8.8.6 Extends

- *<Object>*

8.8.7 Extended By

- *<Container>*
- *<Overlay>*
- <Placemark>

8.9 <Folder >

8.9.1 Syntax

```
<Folder id="ID" targetID="NCName">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

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

8.9.2 Description

A `Folder` is used to organize *`Features`* hierarchically.

8.9.3 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.9.4 Extends

- *<Container>*

8.9.5 Contains

- 0 or more *<Feature>* elements.

8.10 <*Geometry*>

8.10.1 Syntax

```
<!-- abstract element; do not create -->
<!-- Geometry id="ID" targetID="NCName"> -->
                                          <!-- Point,LineString,LinearRing,
                                               Polygon,MultiGeometry,Model -->
<!-- /Geometry -->
```

8.10.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause.

8.10.3 Extends

- <*Object*>

8.10.4 Extended By

- <Point>
- <LineString>
- <LinearRing>
- <Polygon>
- <MultiGeometry>
- <Model>

8.11 <GroundOverlay >

8.11.1 Syntax

```
<GroundOverlay id="ID" targetId="NCName">
  <!-- inherited from Feature element -->
  <name>...</name>                          <!-- string -->
  <visibility>1</visibility>                <!-- boolean -->
  <open>0</open>                            <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                    <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>                 <!-- string -->
  <phoneNumber>...</phoneNumber>            <!-- string -->
  <Snippet maxLines="2">...</Snippet>       <!-- string -->
  <description>...</description>            <!-- string -->
  <AbstractView>...</AbstractView>          <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>        <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>                  <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

  <!-- 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 -->
  <LatLonBox>
    <north>...</north>                      <! kml:angle90 -->
    <south>...</south>                      <! kml:angle90 -->
    <east>...</east>                        <! kml:angle180 -->
    <west>...</west>                        <! kml:angle180 -->
    <rotation>0</rotation>                  <! kml:angle180 -->
  </LatLonBox>
</GroundOverlay>
```

8.11.2 Description

Specifies how to display an image draped over the terrain.

8.11.3 Elements specific to GroundOverlay

<altitude>

Specifies the distance above the terrain in meters. It shall be interpreted according to `altitudeMode`.

<altitudeMode>

Specifies how the `altitude` is interpreted. Possible values are

- **clampToGround** – Indicates to ignore the altitude specification and drape the overlay onto the terrain.
- **absolute** – Sets the altitude of the overlay relative to the vertical datum.

<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 decimal degrees. Values can be ±180, with 0 being 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>
```

8.11.4 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/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>
```

8.11.5 Extends

- *<Overlay>*

8.11.6 Contained By

- <Document>
- <Folder>

8.12 <Icon>

8.12.1 Syntax

```
<Icon id="ID" targetId="NCName">
  <!-- specific to Icon -->
  <href>...</href>                      <!-- anyURI -->
  <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>
```

8.12.2 Description

Defines an image associated with an `IconStyle` or *`Overlay`*. `Icon` has the same content model as `Link`. See <Link>.

8.12.3 Example

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

8.12.4 Contained By

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

8.13 <IconStyle>

8.13.1 Syntax

```
<IconStyle id="ID" targetId="NCName">
  <!-- 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:vec2Type -->
</IconStyle>
```

8.13.2 Description

Specifies how icons for `Placemarks` with a `Point` geometry are drawn in an earth browser's list and geographic views. The color specified in the `color` element of `IconStyle` is blended with the color of the icon.

8.13.3 Elements specific to IconStyle

<scale>

Specifies a scale factor that shall be applied to the icon.

<heading>

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

<Icon>

<href>

A URL specifying the resource location. Relative URLs may be used and are evaluated relative to the enclosing KML instance. See RFC 1808.

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

Specifies the position of the reference point on the icon that is anchored to the `Point` specified in the `Placemark`.

The *x* and *y* values may each 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. They may or may not be specified in a consistent manner — for example, *x* can be specified in pixels and *y* as a fraction.

The origin of the image coordinate system is in the lower left corner of the icon.

The following are attributes of <hotSpot>:

- **x** – The *x* component of the point.
- **y** – The *y* component of the point.
- **xunits** – Units in which the *x* value is specified.
- **yunits** – Units in which the *y* value is specified.

8.13.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.13.5 Extends

- *<ColorStyle>*

8.13.6 Contained By

- <Style>

8.14 <kml>

8.14.1 Syntax

```
<kml xmlns="http://earth.google.com/kml/2.2" hint="...">
  ...
</kml>
```

8.14.2 Description

The root element of a KML document. The optional `hint` attribute may be used to provide information on how to process the KML document instance.

8.14.3 Example

```
<kml xmlns="http://earth.google.com/kml/2.2">
  <NetworkLinkControl> ... </NetworkLinkControl>
  <Document> ... </Document>
</kml>
```

8.15 <LabelStyle>

8.15.1 Syntax

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

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

8.15.2 Description

Specifies how the name of a *Feature* is drawn in the geographic view. A user-defined color, color mode, and scale for the value of the name can be specified.

8.15.3 Elements specific to LabelStyle

<scale>

Specifies a scale factor that shall be applied to the label.

8.15.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.15.5 Extends

- *<ColorStyle>*

8.15.6 Contained By

- <Style>

8.16 <LinearRing>

8.16.1 Syntax

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

8.16.2 Description

Defines a closed line string.

8.16.3 Elements specific to LinearRing

<extrude>

Boolean value. Specifies whether to connect the `LinearRing` to the ground. To extrude this geometry, the `altitudeMode` shall be either **relativeToGround** or **absolute**, and the altitude component within the `coordinates` element should be greater than 0 (that is, in the air). The points of a `LinearRing` are extruded toward the Earth's center of mass.

<tessellate>

Boolean value. Specifies whether to drape the `LinearRing` over the terrain. To enable tessellation, the value for `altitudeMode` shall be **clampToGround**.

<altitudeMode>

Specifies how altitude components in the `coordinates` element are interpreted. Possible values are:

- **clampToGround –** Indicates to ignore an altitude specification.
- **relativeToGround** – Sets the altitude relative to the terrain elevation.
- **absolute –** Sets the altitude relative to the vertical datum.

<coordinates>

Four or more tuples, each consisting of decimal values for geodetic longitude, geodetic latitude, and altitude. The altitude component is optional. The coordinate separator is a comma and the tuple separator is a whitespace. The last coordinate of a `LinearRing` must be the same as the first coordinate. Longitude and latitude coordinates are expressed in decimal degrees only.

8.16.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.16.5 Extends

- <*Geometry*>

8.16.6 Contained By

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

8.17 <LineString>

8.17.1 Syntax

```
<LineString id="ID" targetId="NCName">
  <!-- specific to LineString -->
  <extrude>0</extrude>                  <!-- boolean -->
  <tessellate>0</tessellate>            <!-- boolean -->
  <altitudeMode>clampToGround</altitudeMode>
    <!-- kml:altitudeModeEnum: clampToGround, relativeToGround, or absolute -->
  <coordinates> ... </coordinates>      <!-- lon,lat[,alt] -->
</LineString>
```

8.17.2 Description

Defines a contiguous set of line segments.

8.17.3 Elements specific to LineString

<extrude>

Boolean value. Specifies whether to extend the `LineString` to the ground. To extrude a `LineString`, the value for `altitudeMode` shall be either **relativeToGround** or **absolute**, and at least one altitude component within the `coordinates` element should be greater than 0 (that is, in the air). When a `LineString` is extruded, the points of each line segment are extended to the terrain toward the Earth's center of mass, forming a polygon that looks like a wall or fence.

<tessellate>

Boolean value. Specifies whether to drape the `LineString` over the terrain. To enable tessellation, the value for `altitudeMode` must be **clampToGround**.

<altitudeMode>

Specifies how altitude components in the `coordinates` element are interpreted. Possible values are

- **clampToGround –** Indicates to ignore an altitude specification.
- **relativeToGround** – Sets the altitude relative to the terrain elevation.
- **absolute –** Sets the altitude relative to the vertical datum.

<coordinates>

Two or more coordinate tuples, each consisting of decimal values for geodetic longitude, geodetic latitude, and altitude. The altitude component is optional. The coordinate separator is a comma and the tuple separator is a whitespace. Longitude and latitude coordinates are expressed in decimal degrees only.

8.17.4 Example

```
<kml xmlns="http://earth.google.com/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>0</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>
```

8.17.5 Extends

- *<Geometry>*

8.17.6 Contained By

- <MultiGeometry>
- <Placemark>

8.18 <LineStyle>

8.18.1 Syntax

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

  <!-- specific to LineStyle -->
  <width>1</width>                   <!-- float -->
 </LineStyle>
```

8.18.2 Description

Specifies the drawing style (color, color mode, and line width) for all line geometry. Line geometry includes the `Polygon` boundaries (`LinearRings`) for which the applicable `PolyStyle` outline element value is 1 (true), and lines connecting extruded `Placemarks` with a `Point` geometry to the ground. Use `LineStyle` to specify the color, color mode, and width of the line. For extruded `LineStrings`, the line itself uses the current `LineStyle`, and the extrusion uses the current `PolyStyle`. See also PolyStyle.

8.18.3 Elements specific to LineStyle

<width>

Width of the line, in pixels.

8.18.4 Example

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

```
<kml xmlns="http://earth.google.com/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>
```

8.18.5 Extends

- *<ColorStyle>*

8.18.6 Contained By

- <Style>

8.19 <Link>

8.19.1 Syntax

```
<Link id="ID" targetId="NCName">
  <!-- 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>   </viewFormat>             <!-- string -->
  <httpQuery>...</httpQuery>            <!-- string -->
</Link>
```

8.19.2 Description

`Link` specifies the location of any of the following resources:

- KML documents fetched by `NetworkLink`
- images used by the *`Overlay`* element
- textured 3D objects used by the `Model` element

The resource should be loaded and refreshed according to the refresh parameters supplied. Two different sets of refresh parameters can be specified: one based on time (`refreshMode` and `refreshInterval`) and one based on the current view (`viewRefreshMode` and `viewRefreshTime`). In addition, `Link` specifies whether to scale the bounding box parameters (`viewBoundScale`) and provides a set of optional viewing parameters (`viewFormat`) as well as a set of optional parameters containing version and language information (`httpQuery`).

The valid URL request shall be the concatenation of three pieces of information:

- the `href` (URL) that specifies the resource location
- the `viewFormat` string value used to specify any view parameters
- the `httpQuery` string value used to specify any other query parameters

If the file specified in `href` is a local file, the `viewFormat` and `httpQuery` elements shall be ignored.

NOTE: `Link` should be used instead of the deprecated Url element that it replaces.

8.19.3 Elements specific to Link

<href>

A URL specifying the resource location. Relative URLs may be used and are evaluated relative to the enclosing KML instance. See RFC 1808.

<refreshMode>

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

- **onChange** – refresh when the file is first loaded and whenever the `Link` parameters change.
- **onInterval** – refresh every *n* seconds as 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 RFC2616.

<refreshInterval>

Indicates to refresh the file every *n* seconds.

<viewRefreshMode>

Specifies how the link is refreshed when the geographic view changes.

May be one of the following:

- **never** – Ignore changes in the geographic 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.
- **onRegion** – Refresh the file when the Region becomes active. See `Region`.

<viewRefreshTime>

Specifies the number of seconds to wait before refreshing the geographic view after camera movement stops. This applies when `viewRefreshMode` is set to **onStop**.

<viewBoundScale>

Scales the bounding box parameters. A value less than 1 specifies to use a geographic area less than the current geographic view. A value greater than 1 specifies to use a geographic area greater than the current geographic view.

<viewFormat>

String value used to specify view parameters. The following query parameters may be used:

- **[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 decimal degrees/meters that `LookAt` is viewing
- **[cameraLon]**, **[cameraLat]**, **[cameraAlt]** – decimal 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 geographic view
- **[terrainEnabled]** – indicates whether the geographic view is showing terrain
- **[bboxWest], [bboxSouth], [bboxEast], [bboxNorth]** – bounding box limits matching the OGC Web Map Service (WMS) bounding box specification.

<httpQuery>

String value used to specify any query parameters not related to the geographic view. The following query parameters may be used:

- **[clientVersion] –** version of earth browser client
- **[kmlVersion] –** version of requested kml
- **[clientName] –** name of earth browser client

- **[language]** – language preference of the earth browser client

8.19.4 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>
```

8.19.5 Extends

- *<Object>*

8.19.6 Contained By

- <Model>
- <NetworkLink>

8.19.7 See Also

- <NetworkLinkControl>
- <Region>

8.20 <ListStyle>

8.20.1 Syntax

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

8.20.2 Description

Specifies how a *Feature* is displayed in the list view.

8.20.3 Elements specific to ListStyle

<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, to apply a blue color with 50 percent opacity to an overlay, specify the following: <color>7fff0000</color>, where *alpha*=0x7f, *blue*=0xff, *green*=0x00, and *red*=0x00.

<listItemType>

Specifies how a `Folder` and its contents shall be displayed as items in the list view. Possible values are:

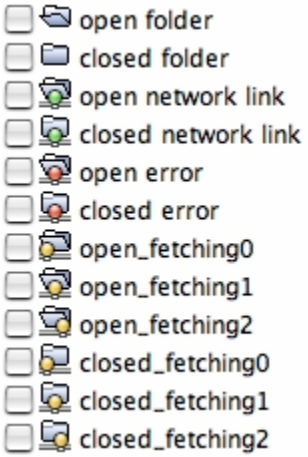
- **check** – The *Feature's* visibility is tied to its item's checkbox.
- **radioFolder** – Only one of the `Folder's` items shall be visible at a time.

- **checkOffOnly** – Prevents all items from being made visible at once—that is, the user can turn everything in the `Folder` off but cannot turn everything on at the same time. This setting is useful for `Folders` containing large amounts of data.

- **checkHideChildren** – Use a normal checkbox for visibility but do not display the `Folder's` children in the list view. A checkbox allows the user to toggle visibility of the child objects in the viewer.

<ItemIcon>

Icon used in the list view that reflects the state of a `Folder` and/or `Link` fetch. Icons associated with the **open** and **closed** modes are used for `Folders`. Icons associated with the **error** and **fetching0**, **fetching1**, and **fetching2** modes are used for `NetworkLinks`. For example, the following screen capture illustrates the possible 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 URL of the image used for the icon in the list view for the *`Feature`*.

8.20.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.20.5 Extends

- *<Object>*

8.20.6 Contained By

- <Style>

8.21 <LookAt>

8.21.1 Syntax

```
<LookAt id="ID" targetId="NCName">
  <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 -->
</LookAt>
```

8.21.2 Description

Specifies the geographic view in terms of a point of interest viewed from a virtual camera. The `LookAt` object is more limited in scope than `Camera` and should establish a view direction that intersects the Earth's surface.

8.21.3 Elements specific to LookAt

<longitude>

Geodetic longitude of the point the camera is looking at. Angular distance in decimal 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>

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

<altitude>

Altitude in meters. See `altitudeMode` for how this value is interpreted.

<heading>

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

<tilt>

Angle, in decimal degrees, 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>

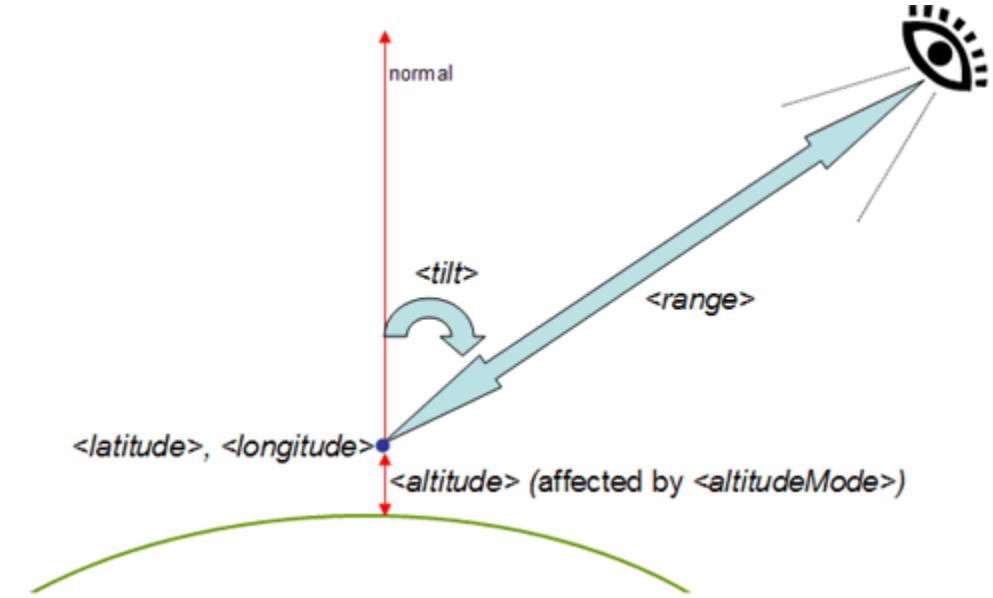
Distance in meters from the point specified by `longitude`, `latitude`, and `altitude` to the `LookAt` position. (See diagram below.)

<altitudeMode>

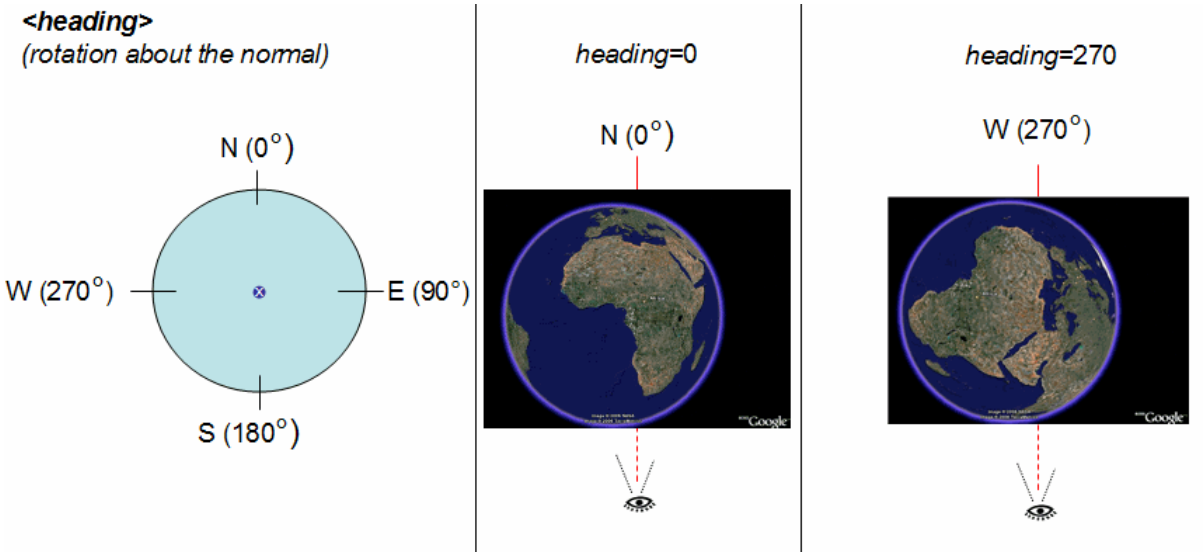
Specifies how the `altitude` specified for the `LookAt` point is interpreted. Possible values are as follows:

- **clampToGround** – Indicates to ignore the `altitude` specification and place the `LookAt` position on the terrain.
- **relativeToGround** – Interprets the `altitude` as a value in meters above the terrain.
- **absolute** – Interprets the `altitude` as a value in meters above the vertical datum.

This diagram illustrates the `range`, `tilt`, and `altitude` elements:



This diagram illustrates the `heading` element:



8.21.4 Example

```
<kml xmlns="http://earth.google.com/kml/2.2">
<Placemark>
  <name>LookAt.kml</name>
  <LookAt>
    <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>
```

8.21.5 Extends

- *<AbstractView>*

8.21.6 Contained By

- *<Feature>*

- <NetworkLinkControl>

8.22 <Model>

8.22.1 Syntax

```
<Model id="ID" targetId="NCName">
  <!-- specific to Model -->
  <altitudeMode>clampToGround</altitudeMode>
    <!--kml:altitudeModeEnum: clampToGround,relativeToGround,or absolute -->
  <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>
```

8.22.2 Description

Specifies the location and orientation of a textured 3D object. The structure and appearance of the textured 3D object is not defined in this specification.

8.22.3 Elements specific to Model

<altitudeMode>

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

- **clampToGround** – Indicates to ignore the `altitude` specification and place the `Model` on the terrain.

- **relativeToGround** – Interprets the `altitude` as a value in meters above the terrain.
- **absolute** – Interprets the `altitude` as a value in meters above the vertical datum.

<Location>

Specifies the coordinates of the `Model's` origin.

<latitude>

Geodetic latitude of origin in decimal degrees.

< longitude >

Geodetic longitude of origin in decimal degrees.

< altitude >

Altitude of origin measured in meters and interpreted according to `altitudeMode`.

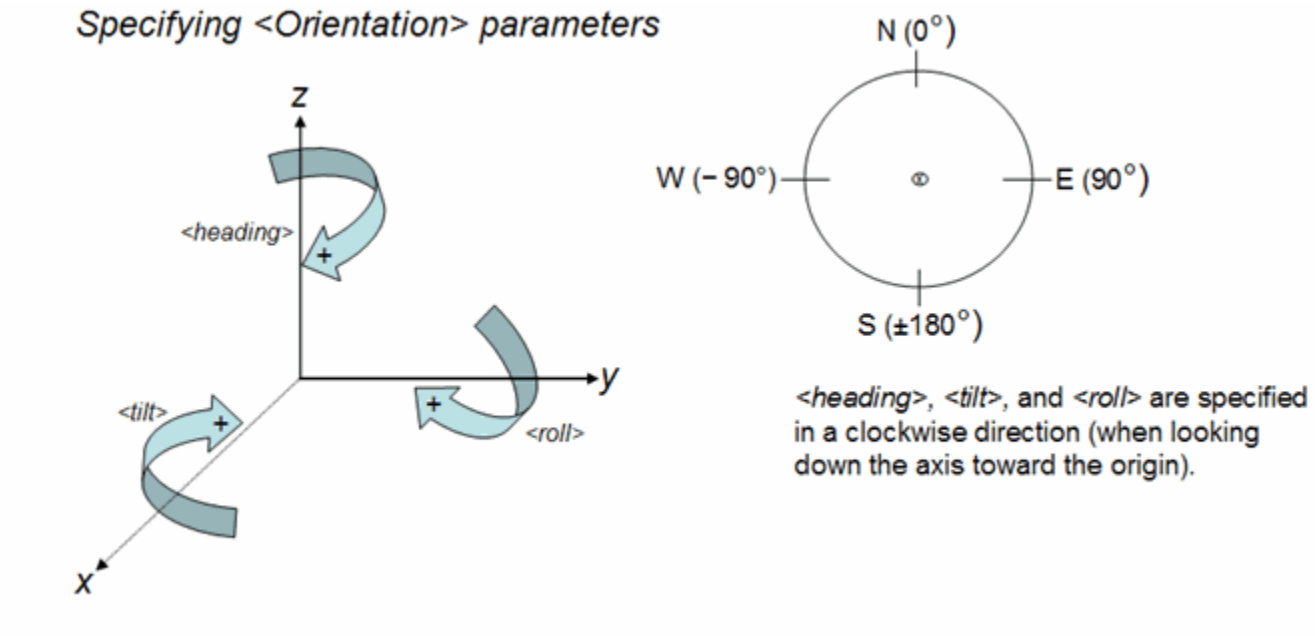
```
<Location>
  <longitude>39.55375305703105</longitude>
  <latitude>-118.9813220168456</latitude>
  <altitude>1223</altitude>
</Location>
```

<Orientation>

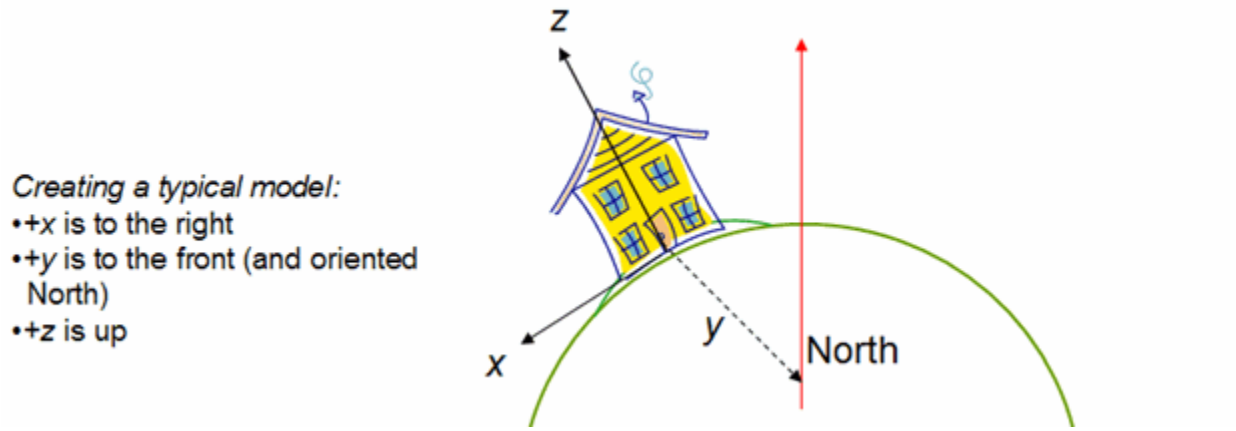
Specifies the orientation of the model coordinate axes relative to a local earth-fixed reference frame. See diagram below.

- **<heading>** – rotation about the *z* axis. A value of 0 equals North. A positive rotation is counter clockwise around the positive *z* axis, looking along the *z*-axis away from the origin, and specified in decimal degrees from 0 to ±180.
- **<tilt>** – rotation about the *x* axis. A positive rotation is counter clockwise around the positive *x* axis and specified in decimal degrees from 0 to ±180.
- **<roll>** – rotation about the *y* axis. A positive rotation is counter clockwise around the positive *y* axis and specified in decimal degrees from 0 to ±180.

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



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



<Scale>

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

<x>

Scale factor along x axis.

<y>

Scale factor along x axis.

<z>

Scale factor along z axis.

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

<Link>

See <Link>.

<ResourceMap>

Specifies 0 or more `Alias` elements, each of which is a mapping for the texture file path from the original textured 3D object file to the KML or KMZ resource that contains the `Model`. This element allows texture files to be moved and renamed without having to update the original textured 3D object file that references those textures. One `ResourceMap` element can contain multiple mappings from different source textured object files into the same target resource.

```
<Alias>
  <targetHref>../images/foo.jpg</targetHref>
  <sourceHref>in-geometry-file/foo.jpg</sourceHref>
</Alias>
```

<Alias>

Contains a mapping from a `sourceHref` to a `targetHref`:

<targetHref>

Specifies the textured 3D object file to be fetched by an earth browser. 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>

Specifies the path for the texture file within the textured 3D object.

8.22.4 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>
```

8.22.5 Extends

- *<Geometry>*

8.22.6 Contained By

- <MultiGeometry>
- <Placemark>

8.23 <MultiGeometry>

8.23.1 Syntax

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

8.23.2 Description

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

8.23.3 Elements specific to MultiGeometry

<Geometry>

See <Geometry>.

8.23.4 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>
```

8.23.5 Extends

- *<Geometry>*

8.23.6 Contained By

- <MultiGeometry>
- <Placemark>

8.24 <NetworkLink>

8.24.1 Syntax

```
<NetworkLink id="ID" targetId="NCName">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

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

8.24.2 Description

References a KML resource on a local or remote network. `NetworkLinks` may be used in combination with `Regions` to efficiently load and display large datasets.

8.24.3 Elements specific to NetworkLink

<refreshVisibility>

Boolean value. Specifies the control over the visibility of any *`Features`* within the referenced KML resource. A value of 0 shall leave the visibility of any referenced *`Features`* in the geographic view within the control of the earth browser user. A value of 1 shall require any referenced *`Features`* to be visible within the geographic view whenever such *`Features`* are refreshed.

<flyToView>

Boolean value. A value of 0 or true indicates that the geographic view shall remain unchanged. A value of 1 or false indicates that the geographic view shall be displayed according to the *AbstractView* specified by either:

- a NetworkLinkControl
- a child *Feature* of <kml>

if they exist in the referenced KML resource. The *AbstractView* of the NetworkLinkControl shall take precedence over the *AbstractView* of the *Feature* if they both exist. If neither exists then the view shall remain unchanged.

<Link>

See <Link>.

8.24.4 Example

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

8.24.5 Extends

- *<Feature>*

8.24.6 Contained By

- *<Container>*

8.25 <NetworkLinkControl>

8.25.1 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>
```

8.25.2 Description

Controls the behaviour of the `NetworkLink` that references the KML resource to which the *`NetworkLinkControl`* belongs. See also <NetworkLink> .

8.25.3 Elements specific to NetworkLinkControl

<minRefreshPeriod>

Specifies in seconds the minimum allowed time between refreshes of the referenced KML resource. The value shall take precedence over the `refreshInterval` element value specified by the `NetworkLink`.

<maxSessionLength>

Specifies in seconds the maximum time that an earth browser shall remain connected to the referenced KML resource. The default value of -1 indicates not to terminate the session explicitly.

<cookie>

Use this element to append a string to the `NetworkLink` URL query.

<message>

Text that should be displayed when a `NetworkLink` is first activated or the `message` value is updated.

<linkName>

Specifies valid content for the `NetworkLink name` element. The `linkName` value shall take precedence over the `NetworkLink name` value.

<linkDescription>

Specifies valid content for the `NetworkLink description` element. The `linkDescription` value shall take precedence over the `NetworkLink description` value.

<linkSnippet maxLines="2" >

Specifies valid content for the `NetworkLink Snippet` element. The `linkSnippet` content shall take precedence over the `NetworkLink Snippet` value.

<expires>

Specifies a point in time at which the `NetworkLink` shall be refreshed. The value shall be formatted according to the dateTime field type. `expires` shall apply only if an associated `Link refreshMode` value is **onExpire**.

<Update>

See <Update>.

<AbstractView>

See *<AbstractView>*.

8.25.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.25.5 Contained By

- <kml>

8.25.6 See Also

- <Update>
- <NetworkLink>

8.26 <*Object*>

8.26.1 Syntax

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

8.26.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause. The optional `id` attribute may be used to specify a unique identifier for the `Object` within a KML instance. It is of XML ID type.

The optional `targetId` attribute may be used to encode the id value of another `Object`. It is of XML NCName type.

8.26.3 Extended By

- <*AbstractView*>
- <BalloonStyle>
- <*ColorStyle*>
- <*Feature*>
- <*Geometry*>
- <Icon>
- <LatLonAltBox>
- <LatLonBox>
- <Link>
- <ListStyle>
- <Location>
- <Lod>

- <Orientation>
- <Region>
- <Scale>
- *<StyleSelector>*
- *<TimePrimitive>*

8.27 *<Overlay>*

8.27.1 Syntax

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

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

8.27.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause.

8.27.3 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, to apply a blue color with 50 percent opacity to an overlay, specify the following: <color>7fff0000</color>

<drawOrder> (default=0)

This element defines the stacking order , relative to the *AbstractView*, for overlapping *Overlays*. *Overlays* with higher drawOrder values are drawn on top of overlays with lower drawOrder values.

<Icon>

Specifies the image associated with the *Overlay*. If no image is specified or located, a rectangle is drawn using the color and size defined by the ground or screen overlay. See <Icon>.

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

8.27.4 Extends

- *<Feature>*

8.27.5 Extended By

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

8.28 <PhotoOverlay>

8.28.1 Syntax

```
<PhotoOverlay id="ID" targetId="NCName" -->
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

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

  <!-- specific to PhotoOverlay -->
  <shape>rectangle</shape>              <!-- kml:shape -->
  <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>
  <roll>0</roll>                        <!-- kml:angle180 -->
  <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>
</PhotoOverlay>
```

8.28.2 Description

The PhotoOverlay element allows a photograph to be geographically located relative to the Earth and viewing parameters to be specified for this PhotoOverlay. The PhotoOverlay may be rendered as 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.

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).

8.28.3 Elements Specific to PhotoOverlay

<shape>

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

- **rectangle** – for an ordinary photo
- **cylinder** – for panoramas, which can be either partial or full cylinders
- **sphere** – for spherical panoramas

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

<rightFov>

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

<bottomFov>

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

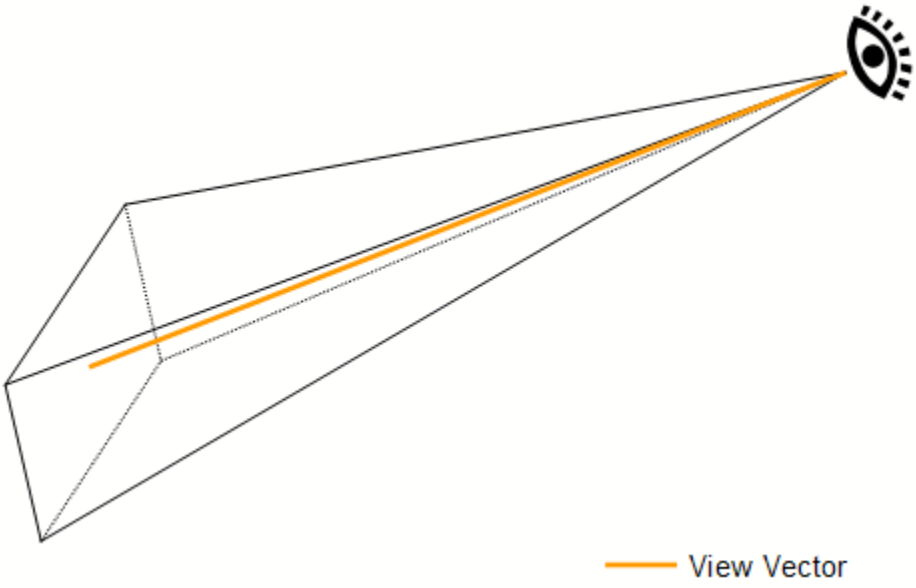
<topFov>

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

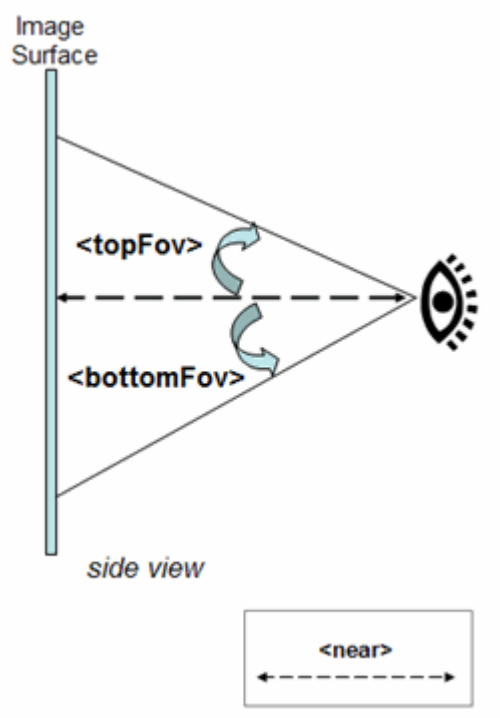
<near>

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

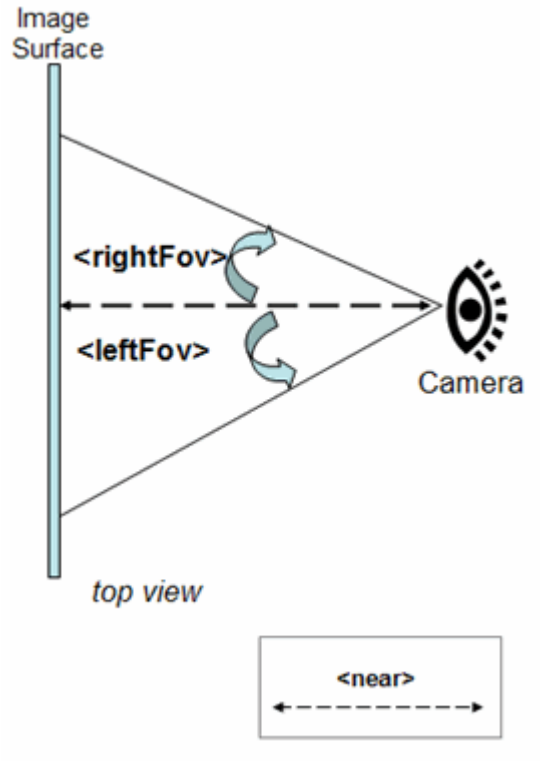
The field of view for a `PhotoOverlay` is defined by four planes, each of which is specified by an angle relative to the *view vector*. These four planes define the top, bottom, left, and right sides of the field of view, which has the shape of a truncated pyramid, as shown here:



Here is the side view of the field of view, which shows the `topFov` and `bottomFov` angles:



And here is the top view of the field of view, which shows the `leftFov` and `rightFov` angles:



<roll>

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

<ImagePyramid>

An image pyramid may be specified. This is a hierarchical set of images, each of which is an increasingly lower resolution (towards the top of the pyramid). Each image in the pyramid is subdivided into tiles so only the portions in view are loaded.

When an image pyramid is specified, the <href> in the <Icon> element shall be modified 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>

An `Icon` may be used to style the `Point`. The `Icon` is specified by the `styleUrl` and `StyleSelector` fields, just as it is for `Placemark`. See <Point>.

8.28.4 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: <shape> -->
  <ViewVolume>
    <near>1000</near>
    <leftFov>-60</leftFov>
    <rightFov>60</rightFov>
    <bottomFov>-45</bottomFov>
    <topFov>45</topFov>
  </ViewVolume>
  <!-- default: <roll> default is 0 -->
  <Point>
    <coordinates>1,1</coordinates>
  </Point>
  <!-- if no ImagePyramid only level 0 is shown,
       fine for a non-pyramidal image -->
</PhotoOverlay>
```

8.28.5 Extends

- *<Overlay>*

8.28.6 Contained By

- <Folder>
- <Document>
- <kml>

8.29 <Placemark>

8.29.1 Syntax

```
<Placemark id="ID" targetId="NCName" -->
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

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

8.29.2 Description

A `Placemark` is a *`Feature`* with associated *`Geometry`*.

A `Placemark` with a `Point` geometry should be drawn with an icon to mark the `Placemark` in the geographic view. The point itself determines the position of the `Placemark's` name and display icon. See <IconStyle>.

8.29.3 Elements specific to Placemark

<Geometry>

See <Geometry>.

8.29.4 Example

```
<Placemark>
  <name>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>
```

8.29.5 Extends

- *<Feature>*

8.29.6 Contained By

- <Document>
- <Folder>

8.29.7 See Also

- <Icon>

8.30 <Point>

8.30.1 Syntax

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

8.30.2 Description

A geographic location defined by longitude, latitude, and (optional) altitude.

8.30.3 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` shall be either **relativeToGround** or **absolute**, and the altitude component within the `coordinates` element should be greater than 0 (that is, in the air). The point is extruded toward the Earth's center of mass.

<altitudeMode>

Specifies how altitude components in the `coordinates` element are interpreted. Possible values are

- **clampToGround –** Indicates to ignore an altitude specification.
- **relativeToGround** – Sets the altitude relative to the terrain elevation.
- **absolute –** Sets the altitude relative to the vertical datum.

<coordinates>

A single coordinate tuple consisting of decimal values for geodetic longitude, geodetic latitude, and altitude. The altitude component is optional. The coordinate separator is a comma. Longitude and latitude coordinates are expressed in decimal degrees only.

8.30.4 Example

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

8.30.5 Extends

- *<Geometry>*

8.30.6 Contained By

- <MultiGeometry>
- <Placemark>

8.31 <Polygon>

8.31.1 Syntax

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

8.31.2 Description

A `Polygon` is defined by an outer boundary and 0 or more inner boundaries. Each boundary is defined by a `LinearRing`. See <LinearRing>.

8.31.3 Elements specific to Polygon

<extrude>

Boolean value. Specifies whether to connect the `Polygon` to the ground. To extrude a `Polygon`, the value for `altitudeMode` shall be either **relativeToGround** or **absolute**, and the altitude component within the `coordinates` element should be greater than 0 (that is, in the air). When a `Polygon` is extruded, each boundary point is extended to the terrain toward the earth's center of mass, which gives the appearance of a building or a box. Only the `Polygon` boundary is extruded, not the `Polygon` interior (for example, a rectangle turns into a box with five faces). The boundary points of the `Polygon` are extruded toward the Earth's center of mass.

<tessellate>

Boolean value. Specifies whether to drape the `Polygon` over the terrain. To enable tessellation, the `Polygon` must have an `altitudeMode` of **clampToGround**.

<altitudeMode>

Specifies how altitude components in the `coordinates` element are interpreted. Possible values are

- **clampToGround –** Indicates to ignore an altitude specification.
- **relativeToGround** – Sets the altitude relative to the terrain elevation.
- **absolute –** Sets the altitude relative to the vertical datum.

<outerBoundaryIs>

Specifies the exterior boundary of the `Polygon` defined by a `LinearRing` element.

<innerBoundaryIs>

Specifies an inner boundary of the `Polygon` defined by a `LinearRing` element.

8.31.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.31.5 Extends

- *<Geometry>*

8.31.6 Contained By

- <MultiGeometry>
- <Placemark>

8.32 <PolyStyle>

8.32.1 Syntax

```
<PolyStyle id="ID" targetId="NCName">
  <!-- 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>
```

8.32.2 Description

Specifies the drawing style for a polygon, including a `Polygon` and the extruded portion of a `Polygon` or `LineString`.

8.32.3 Elements specific to PolyStyle

<fill>

Boolean value. Specifies whether to fill the polygon.

<outline>

Boolean value. Specifies whether to outline the polygon.

NOTE: Polygon outlines use the current LineStyle.

8.32.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.32.5 Extends

- *<ColorStyle>*

8.32.6 Contained By

- <Style>

8.33 <Region>

8.33.1 Syntax

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

8.33.2 Description

A `Region` affects visibility of a *`Feature`*. `Regions` define both culling and level-of-detail control over the display of the *`Feature`*. A region shall specify a `LatLonAltBox` element that describes an area of interest defined by geographic coordinates and altitudes. In addition, a `Region` contains a `Lod` element that defines a validity range of the associated `Region` in terms of projected screen size. See <LatLonAltBox> and <Lod>.

Regions are inherited through a *`Feature`* hierarchy and affect the visibility of *`Features`* that are defined lower in the hierarchy.

A `Region` is said to be "active" when the bounding box is within the user's view and the LOD requirements are met. *`Features`* 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. In a *`Container`* or `NetworkLink` hierarchy, this calculation uses the `Region` that is the closest ancestor in the hierarchy.

8.33.3 Elements specific to Region

<LatLonAltBox>

A bounding box that describes an area of interest defined by geographic coordinates and

altitudes. Default values and required fields are as follows:

<altitudeMode>

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

- **clampToGround** – Indicates to ignore the `minAltitude` and `maxAltitude` specification.
- **relativeToGround** – Interprets the the `minAltitude` and `maxAltitude` as a value in meters above the terrain.
- **absolute** – Interprets the the `minAltitude` and `maxAltitude` as a value in meters above the vertical datum.

<minAltitude>

Specified in meters above the vertical datum (and is affected by the `altitudeMode` specification).

<maxAltitude>

Specified in meters above the vertical datum (and is affected by the `altitudeMode` specification).

<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.

<west>

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` 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.

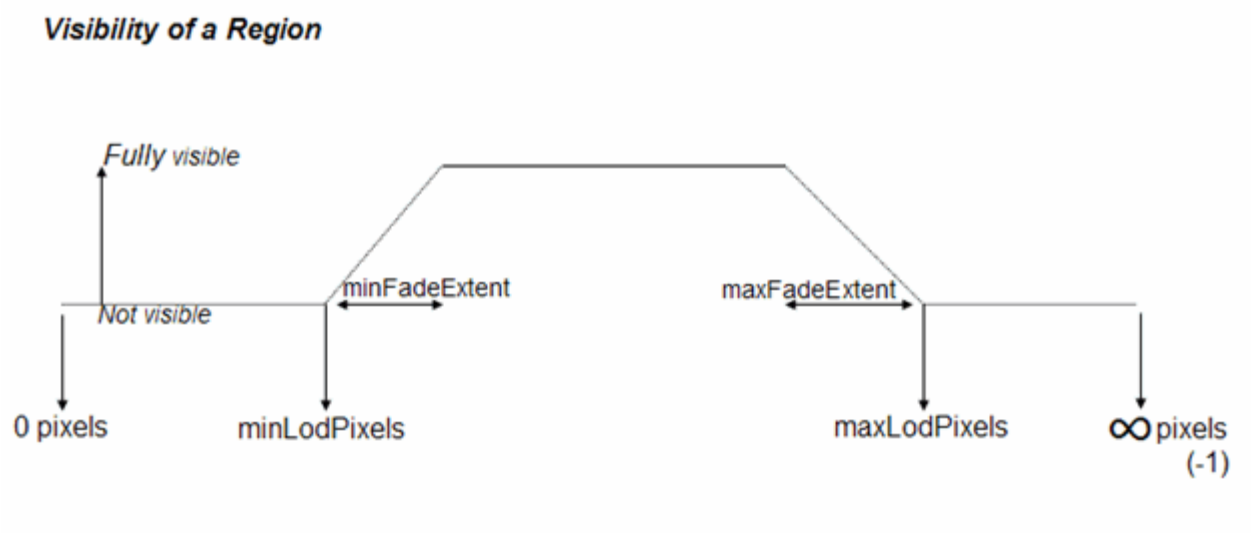
```
<Lod>
  <minLodPixels>256</minLodPixels>
  <maxLodPixels>-1</maxLodPixels>
  <minFadeExtent>0</minFadeExtent>
  <maxFadeExtent>0</maxFadeExtent>
</Lod>
```

- **<minLodPixels>** Measurement in screen pixels that represents the minimum limit of the visibility range for a given `Region`. An earth browser should calculate the size of the `Region` when projected onto screen space then compute the square root of the `Region's` area. For example, if an untiled `Region` is square and the viewpoint is directly above the `Region`, 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, then the `Region` should be activated. If this limit is not reached, the associated geometry should not be drawn since it would be too far from the user's viewpoint to be visible.

- **<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."

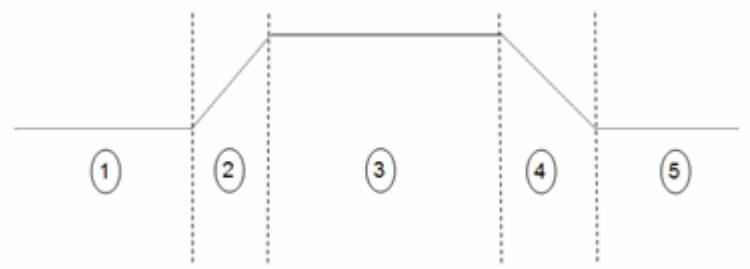
- **<minFadeExtent>** Distance over which the geometry fades, from fully opaque to fully transparent. This ramp value, expressed in screen pixels, is applied at the minimum end of the LOD (visibility) limits.

- **<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
```



8.33.4 Example

```
<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>
```

8.33.5 Extends

- *<Object>*

8.33.6 Contained By

- *<Feature>*

8.34 <Schema>

8.34.1 Syntax

```
<Schema name="string" targetId="NCName">
  <SimpleField type=string" name="string">
    <displayName>...</displayName>            <!-- string -->
  </SimpleField>
</Schema>
```

8.34.2 Description

Specifies a user-defined schema that is used to add user-defined data encoded within a *Feature* `ExtendedData` element. The `id` attribute is required.

A `Schema` may contain one or more `SimpleField` elements.

8.34.3 Elements Specific to Schema

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

Specifies a user-defined field. The `name` attribute specifies the name of the field. The `type` attribute specifies the type of the field. The `type` may be one of the following XML Schema types:

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

<displayName>

An alternate display name for field. `displayName`.

8.34.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.34.5 Contained By

- <Document>

8.34.6 See Also

- <SchemaData>

8.35 <ScreenOverlay>

8.35.1 Syntax

```
<ScreenOverlay targetId="NCName">
  <!-- inherited from Feature element -->
  <name>...</name>                      <!-- string -->
  <visibility>1</visibility>            <!-- boolean -->
  <open>0</open>                        <!-- boolean -->
  <atom:author xmlns="http://www.w3.org/2005/Atom">...<atom:author>
  <atom:link xmlns="http://www.w3.org/2005/Atom">...</atom:link>
  <address>...</address>                <!-- string -->
  <xal:AddressDetails xmlns="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0">...
      </xal:AddressDetails>             <!-- string -->
  <phoneNumber>...</phoneNumber>        <!-- string -->
  <Snippet maxLines="2">...</Snippet>   <!-- string -->
  <description>...</description>        <!-- string -->
  <AbstractView>...</AbstractView>      <!-- LookAt or Camera -->
  <TimePrimitive>...</TimePrimitive>    <!-- TimeStamp or TimeSpan -->
  <styleUrl>...</styleUrl>              <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <ExtendedData>...</ExtendedData>

  <!-- 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"/>
    <!-- vec2Type -->
    <!-- xunits and yunits can be one of: fraction, pixels, or insetPixels -->
  <screenXY x="double" y="double" xunits="fraction" yunits="fraction"/>
    <!-- vec2Type -->
  <rotationXY x="double" y="double" xunits="fraction" yunits"fraction"/>
    <!-- vec2Type -->
  <size x="double" y="double" xunits="fraction" yunits="fraction"/>
    <!-- vec2Type -->
  <rotation>0</rotation>                   <!-- float -->
</ScreenOverlay>
```

8.35.2 Description

Specifies an image overlay that shall be displayed fixed to the screen. `ScreenOverlay` sizing is determined using 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`. The image may be rotated by `rotation` degrees about a point relative to the screen specified by `rotationXY`.

8.35.3 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 –** The *x* component of the point.
- **y** – The *y* component of the point.
- **xunits** – Units in which the *x* value is specified. If it is not specified then the default value of fraction shall apply.
- **yunits** – Units in which the *y* value is specified. If it is not specified then the default value of fraction shall apply.

<screenXY>

Specifies a point relative to the screen origin that the overlay image is mapped to. 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 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** – The *x* component of the point.
- **y** – The *y* component of the point.
- **xunits** – Units in which the *x* value is specified. If it is not specified then the default value of fraction shall apply.
- **yunits** – Units in which the *y* value is specified. If it is not specified then the default value of fraction shall apply.

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

Usage examples:

To 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>
```

To 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>
```

To 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.

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** – The *x* component of the point.
- **y** – The *y* component of the point.
- **xunits** – Units in which the *x* value is specified. If it is not specified then the default value of fraction shall apply.
- **yunits** – Units in which the *y* value is specified. If it is not specified then the default value of fraction shall apply.

<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:

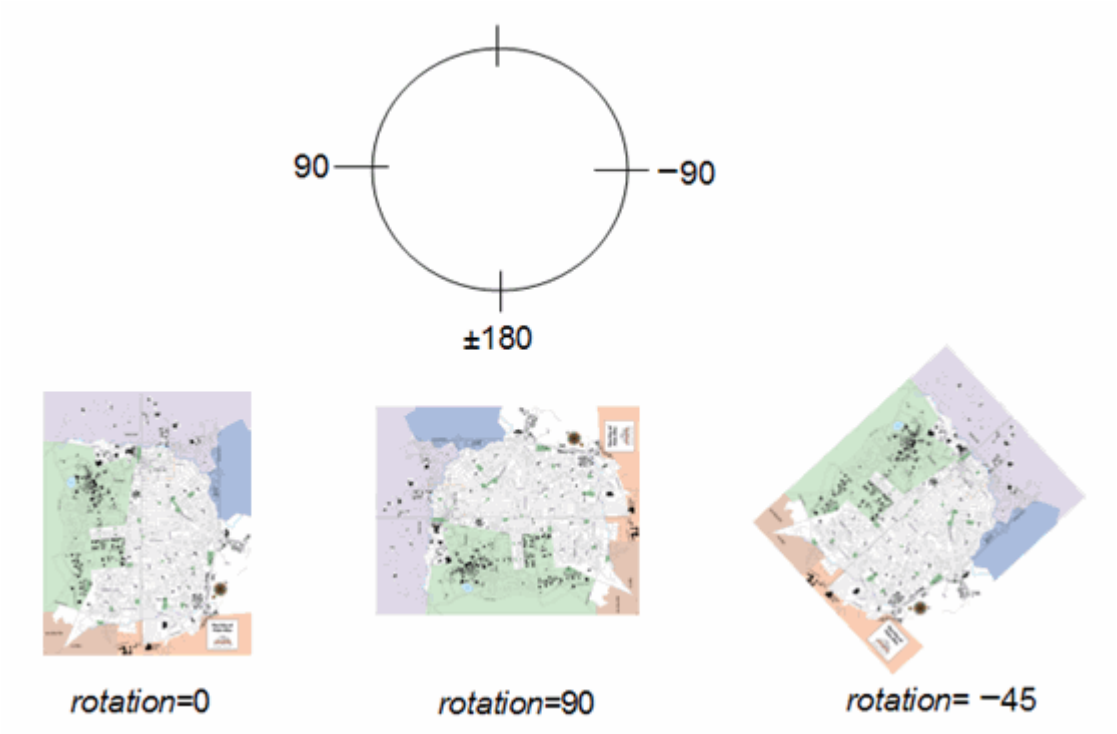
```
<size x="-1" y="0.2" xunits="fraction" yunits="fraction"/>
```

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, in decimal degrees, of the parent object. A value of 0 means no rotation. The value is an angle in decimal 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>.



8.35.4 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>
```

8.35.5 Extends

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

8.35.6 Contained By

- <Document>
- <Folder>

8.36 <Style>

8.36.1 Syntax

```
<Style id="ID" targetId="NCName">
<!-- specific to Style -->
  <IconStyle>...</IconStyle>
  <LabelStyle>...</LabelStyle>
  <LineStyle>...</LineStyle>
  <PolyStyle>...</PolyStyle>
  <BalloonStyle>...</BalloonStyle>
  <ListStyle>...</ListStyle>
</Style>
```

8.36.2 Description

Specifies a container of zero or more `ColorStyles` that can referenced from a `StyleMap` or *`Feature`*. Styles affect how a *`Geometry`* is presented in the geographic view and how a *`Feature`* appears in the list view.

8.36.3 Elements specific to Style

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

8.36.4 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>
```

8.36.5 Extends

- *<StyleSelector>*

8.36.6 Contained By

- *<Feature>*

8.37 <StyleMap>

8.37.1 Syntax

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

8.37.2 Description

Specifies a mapping between two `Styles`. A `StyleMap` may be used to provide separate normal and highlighted styles for a `Placemark`.

8.37.3 Elements specific to StyleMap

<Pair>

Defines a key/value pair that maps a mode to the predefined `styleUrl`. `Pair` contains the following two required elements:

- `key`, which identifies a key whose value is either **normal** or **highlighted**.
- `styleUrl`, which references the style. For referenced style elements that are local to the KML instance, 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/population.xml#example_style_off</styleUrl>
</Pair>
```

8.37.4 Example

```
<kml xmlns="http://earth.google.com/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>
```

8.37.5 Extends

- *<StyleSelector>*

8.37.6 Contained By

- any *<StyleSelector>*

8.38 <*StyleSelector*>

8.38.1 Syntax

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

8.38.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause.

8.38.3 Extends

- *<Object>*

8.38.4 Extended By

- <Style>
- <StyleMap>

8.39 <*TimePrimitive*>

8.39.1 Syntax

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

8.39.2 Description

This is an abstract element that is extended by the elements listed in the Extended By sub clause. Time values encoded within elements that extend TimePrimitive shall be in the context of the temporal reference system specified by ISO 8601, which uses the Gregorian Calendar and 24 hour local or Coordinated Universal Time (UTC).

8.39.3 Extends

- <Object>

8.39.4 Extended By

- <TimeSpan>
- <TimeStamp>

8.40 <TimeSpan>

8.40.1 Syntax

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

8.40.2 Description

Specifies an extent in time bounded by begin and end temporal values. At least one of the child elements `begin` and `end` shall be encoded.

8.40.3 Elements specific to TimeSpan

<begin>

Describes the beginning instant of a time period. If absent, the beginning of the period is unbounded. The value shall be encoding according to the `dateTimeType` field type. See `dateTimeType`.

<end>

Describes the ending instant of a time period. If absent, the end of the period is unbounded. The value shall be later than the `begin` value. The value shall be encoding according to the `dateTimeType` field type. See `dateTimeType`.

8.40.4 Example

The following example shows the time period representing Colorado's statehood. It contains only a `begin` element 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>
```

8.40.5 Extends

- *<TimePrimitive>*

8.40.6 Contained By

- *<Feature>*

8.41 <TimeStamp>

8.41.1 Syntax

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

8.41.2 Description

Specifies a single moment in time.

8.41.3 Elements specific to TimeStamp

<when>

Specifies a single moment in time. The value shall be encoded according to the `dateTimeType` field type. See `dateTimeType`.

The following examples show different temporal 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>
```

8.41.4 Extends

- *<TimePrimitive>*

8.41.5 Contained By

- *<Feature>*

8.42 <Update>

8.42.1 Syntax

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

8.42.2 Description

Specifies an addition, change, or deletion to a KML resource that has previously been retrieved via `NetworkLink`.

NOTE: Update does not affect the KML resource itself; rather it updates its representation within the earth browser only.

8.42.3 Elements specific to Update

May contain zero or more `Change`, `Create`, and `Delete` elements, encoded in any order. They shall be processed in document order.

<targetHref>

Specifies the URL for the target KML resource that has been previously retrieved via `NetworkLink`.

<Change>

Specifies modifications to zero or more identified *`Object`*, *`Feature`*, *`Geometry`*, *`StyleSelector`*, *`TimePrimitive`*, and *`AbstractView`* elements in the target resource.

Target elements to be modified are identified as children of the `Change` element and must include the `targetId` attribute. Modifications to the identified *`Object`* are specified by the content of these children.

The content of identified target elements not subject to modification shall remain unchanged.

<Create>

Specifies the addition of zero or more *Features* to an identified Folder or Document in the target resource.

The Folder or Document shall be identified as a child of the Create element and must include the targetId attribute. New *Features* to be added to the identified Folder or Document are specified as the content of this child.

The targetHref for a created *Feature* is the same as that of the target KML resource.

<Delete>

Specifies the deletion of zero or more *Features* in the target resource.

Features shall be identified as children of the Create element and must include the targetId attribute.

8.42.4 Examples

8.42.4.1 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>
```

8.42.4.2 Example of <Create>

This example creates a new Placemark in a previously created Document that has an id of region24. Note that to make subsequent updates to placemark891, http://myserver.com/Point.kml is used as the targetHref value.

```
<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>
```

8.42.4.3 Example of <Delete>

This example deletes a `Placemark` previously loaded into an earth browser. 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"></>
  </Delete>
</Update>
```

8.42.5 Contained By

- <NetworkLinkControl>

Annex A
(informative)

KML Coordinate Reference System Dictionary

```
<Dictionary gml:id="KMLCRSandUOMDictionary" xmlns="http://www.opengis.net/gml/3.2"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gml/3.2
gml3.2.1/ISO_19136_Schemas/gml.xsd">
  <identifier codeSpace="urn:x-
ogc:def:dictionary:OGC:0.1:KMLCRSandUOMDictionary">KMLCRSDictionary</identifier>
  <dictionaryEntry>
    <CompoundCRS gml:id="LonLat84_5773">
      <identifier codeSpace="urn:x-
ogc:def:cs:OGC:0.1:LonLat84_5773">LonLat84_5773</identifier>
      <name>Geographic 3D Long(deg)/Lat(deg)/Elev(m)</name>
      <scope>KML 3D spatial framework corresponding to absolute
altimodeMode.</scope>
      <componentReferenceSystem xlink:href="#LonLat84" xlink:title="WGS 84 with
long/lat order (deg)"/>
      <componentReferenceSystem xlink:href="#EGM96GeiodHeight" xlink:title="EGM96
geoid"/>
    </CompoundCRS>
  </dictionaryEntry>
  <dictionaryEntry>
    <GeodeticCRS gml:id="LonLat84">
      <identifier codeSpace="urn:x-
ogc:def:cs:OGC:0.1:LonLat84">LonLat84</identifier>
      <name>WGS 84 with long/lat order (deg)</name>
      <scope>KML geographic 2D spatial framework</scope>
      <usesEllipsoidalCS xlink:href="#LonLatEllipsoidalCS" xlink:title="Ellipsoidal
2D CS lon/lat (deg)"/>
      <usesGeodeticDatum xlink:href="urn:x-ogc:def:datum:EPSG:6.12:6326"
xlink:title="World Geodetic System 1984"/>
    </GeodeticCRS>
  </dictionaryEntry>
  <dictionaryEntry>
    <EllipsoidalCS gml:id="LonLatEllipsoidalCS">
      <identifier codeSpace="urn:x-
ogc:def:cs:OGC:0.1:LonLatEllipsoidalCS">LonLatEllipsoidalCS</identifier>
      <name>Ellipsoidal 2D CS lon/lat (deg)</name>
      <usesAxis xlink:href="urn:x-ogc:def:axis:EPSG:6.12:9902" xlink:title="Long"/>
      <usesAxis xlink:href="urn:x-ogc:def:axis:EPSG:6.12:9901" xlink:title="Lat"/>
    </EllipsoidalCS>
  </dictionaryEntry>
  <dictionaryEntry>
    <VerticalCRS gml:id="EGM96GeiodHeight">
      <identifier codeSpace="urn:x-
ogc:def:crs:OGC:0.1:EGM96GeiodHeight">EGM96GeiodHeight</identifier>
      <name>EGM96 geoid</name>
      <remarks>Height surface resulting from the application of the EGM96 geoid
model to the WGS 84 ellipsoid.</remarks>
      <scope>Geodesy.</scope>
```

```
      <verticalCS xlink:href="urn:x-ogc:def:cs:EPSG:6.12:6499"
xlink:title="Gravity-related CS. Axis: height (H). Orientation: up.  UoM: m."/>
      <verticalDatum xlink:href="urn:x-ogc:def:datum:EPSG:6.12:5171"
xlink:title="EGM96 geoid"/>
    </VerticalCRS>
  </dictionaryEntry>
</Dictionary>
```

Bibliography

Google Inc., *KML 2.2 Reference Documen*t, 2007
http://code.google.com/apis/kml/documentation/kml_tags_beta1.html

IETF RFC 1808, *Relative Uniform Resource Locators*. (June 1995)

IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*. (August 1998)

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*. (June 1999)

IETF RFC 2806, *URLs for Telephone Calls*. (April 2000)

IETF RFC 4287, *Atom Syndication Format*: http://atompub.org.

ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*.

ISO 19107:2003, *Geographic Information — Spatial schema*.

ISO 19111:—1), *Geographic Information — Spatial referencing by coordinates*.

OGC document 00-014r1, *Guidelines for Successful OGC Interface Specifications*.

OGC document 99-100r1, *Abstract Specification Topic 0: Overview*.

W3C *HTML and URLs*: http://www.w3.org/TR/WD-html40-970708/htmlweb.html

W3C XML, *Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation*. (4 February 2004)

W3C XML Namespaces, *Namespaces in XML. W3C Recommendation*. (14 January 1999)

W3C XML Schema Part 1, *XML Schema Part 1: Structures. W3C Recommendation*. (2 May 2001)

W3C XML Schema Part 2, *XML Schema Part 2: Datatypes. W3C Recommendation*. (2 May 2001)