NASA WorldWind

WorldWind Markup Language WWML v0.93

(review version)

Authors:

Frank Kuehnel, (USRA) RIACS Nasa Ames

David Burggraf, Galdos Systems, Inc.

Contributors:

Patrick Hogan, Ronald Lake, Chris Maxwell, Randy Kim

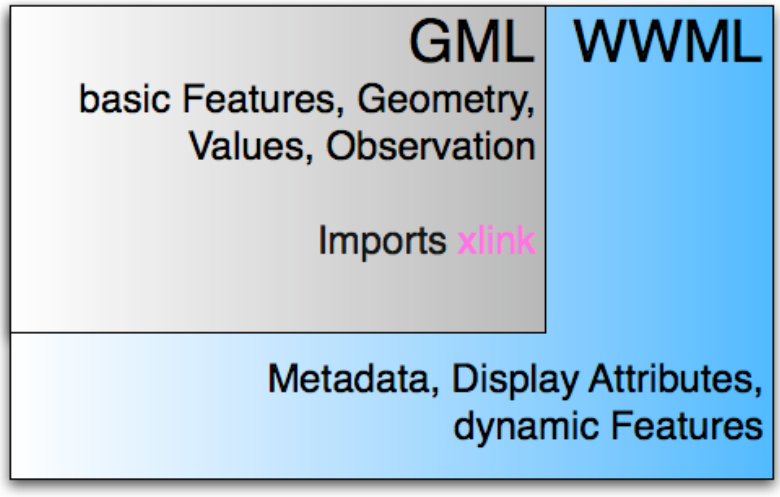
Table of contents

Introduction

WorldWind markup language (WWML) is a specification for interactive and dynamic geo-spatial & temporal data import/export into the visual standalone browser WorldWind or any WWML enabled web-browser written in XML. WWML is a NASA Open Source Standard, its purpose is to precisely describe what and how geospatial & temporal information is displayed in a WorldWind type viewer.

WWML is based on GML 3.1 but also extends the scope in regards to display style options, dynamic features, other supported legacy data files and metadata, hence the “gml” namespace imported by the “wwml” namespace! One of the requirements in development of GML 3.1 (and previous version) was to enable the separation of data and presentation. Styling GML data is typically handled by separate mechanisms although GML default styles can be utilized. GML 3.1 is a rich language specification and a full implementation at this time in WorldWind is not required. We have chosen a suitable profile of GML 3.1 as the basis for WWML.

Further, as GML is XML, geographic information encoded in GML tends to be verbose, thus using precious bandwidth transmission resources & CPU ingestion time. WWML recognizes these limitations. In an example, shapefiles that were converted into GML via “*ogr2ogr*” occupy roughly three times more storage space. Even after compression with “*gzip*” the difference still is a factor of 1.7 in disfavor of the compressed GML (GMZ, WWMZ). Therefore WWML will support binary compressed XML files BXML proposed by CubeWerx and currently implemented in and open-source C-language library for parsing and generating XML and BXML formats under the GNU LGPL license. The source code package can



be found at:http://www.cubewerx.com/main/cwxml Further, legacy data formats, i.e. shapefiles are supported in the WWML specification.

Another area where GML is versatile is metadata. Thus the resulting WWML data descriptor is a mixture of “gml:” and “wwml:” namespaces. It is suggested that “wwml:” namespace can be used in substitution for the “gml:” one for use within WorldWind standalone application.

WWML can not only describe geospatial data resources such as files that contain geographic referenced images, it also can be embedded into a other XML documents, i.e. HTML/XHTML. Therefore, very rich XHTML documents can be designed for future web-browsers that enable WorldWind type rendering and interaction capabilities.

WWML is a NASA open specification standard that is aimed at becoming the defacto HTML/XHTML extension standard for streaming geospatial data over the internet. WWML will be leveraged in the standalone geobrowser WorldWind and in possible future web-browsers that integrate WorldWind type capabilities. The WWML extension shall be seen analogous to the SVG specification.

Defining an WWML document fragment

A **WWML document fragment** consists of any number of WWML/GML elements within an ‘wwml’ element.

An WWML document fragment can stand by itself as a self contained file, in which case it would describe a data resource used in a standalone WWML compliant browser such as WorldWind, or it can be embedded inline as a fragment within a parent XML document. Embedding a WWML fragment in a parent XHTML content would allow a WWML enabled browser to view text and graphics content along with geographic display content ala WorldWind in a predefined window screen on the overall web page.

The following example shows simple WWML content embedded inline as a fragment within a parent XML document. The WWML content is shown in a 8cm wide and 8cm high window within the XHMTL page. Note the use of XML namespaces to indicate that the ‘wwml’ elements and ‘gml’ belong to WWML and respective GML namespaces:

```
<?xml version="1.0" encoding="UTF-8" standalone=”yes” ?>

<parent xmlns="http://www.examples.org" xmlns:wwml="http://worldwind.nasa.gov/wwml”
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink" >

<!-- parent contens here, any XHTML website description... -->

<wwml:wwml width=”8cm” height=”8cm” version=”1.0” lockview=”yes”>

        <!-- specify WWML content here, demonstrated below -->

</wwml:wwml>

<!-- more parent content here ... -->

</parent>
```

Analogously, WWML as a standalone data resource descriptor is shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone=”no” ?>
```

```
<wwml xmlns:wwml="http://worldwind.nasa.gov/wwml" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" >

        <!-- specify WWML content here, demonstrated below -->

</wwml>
```

WWML content structure

The embedding context of a WWML document was presented in the previous section. WWML can stand alone as a resource descriptor for geographic data or as an embedded XML code segment within an HTML/XHTML document. This section presents the general structure of the WWML content, that is the XML code segment between the <wwml> and </wmml> tags.

Although, the WWML/GML structure is versatile and particularly allows for reordering of its elements within the GML grammar, it is useful to think of WWML content in three main segments. Those segments are firstly Metadata, secondly Coordinate Systems & Camera View definitions and thirdly the geographic data features.

```
<wwml>
        <!-- Metadata -->

        <!-- Coordinate system & Preset Camera Views -->

        <!-- styles, geographic layers & features -->
</wwml>
```

Each of the above mentioned sections will be explained in detail.

Metadata
Metadata describes the context of numerical data in general and geographical data in particular. In its simplest form it may consist of title, author(s), publisher(s), keywords. Numerous—but few well known—metadata models exist for geographic objects, ISO/TC211 19115, and the CSDGM/FGDC. The OpenGeospatial consortium recommended specification of metadata [1] follows the ISO19115 recommendation. The ISO19115 standard does not specify the storage format. An XML schema implementation ISO19139 of the ISO19115 standard is currently under development. WWML implements a minimal metadata XML schema set from the ISO19139 specification.

The mandatory (ISO/DIS19115) core metadata elements are

- Dataset title
- Dateset reference date
- Dataset language
- Dataset topic category
- Abstract describing the dataset
- Dataset quality
- Metadata point of contact
- Metadata date stamp

```
<MD_Metadata id=”root”>

        <smXML:language>
                <smXML:CharacterString> en </smXML:CharacterString>
        </smXML:language>
        <smXML:dateStamp>
                <smXML:Date>2006-13-12</smXML:Date>
        </smXML:dateStamp>
        <smXML:identificationInfo>
                <smXML:title>
                        <smXML:CharacterString>
                                Earth from above
                        </smXML:CharacterString>
                </smXML:title>
                <smXML:abstract>
                        <smXML:CharacterString>
                                dataset describes ...
                        </smXML:CharacterString>
                </smXML:abstract>
                <smXML:descriptiveKeywords>
                        <smXML:MD_Keywords>
                                <smXML:keyword>
                                <smXML:CharacterString> Landsat <smXML:CharacterString>
                                </smXML:keyword>
                                ...
                        <smXML:MD_Keywords>
                </smXML:descriptiveKeywords>
        </smXML:identificationInfo>

        <dataQualityInfo>
        </dataQualityInfo>
</MD_Metadata>
```

This section needs more elaboration, to my knowledge only very few documented examples were available for download of the internet!?

Coordinate Systems and preset Camera Views

When referring to geographic locations, the pair of numbers encoding the geographic location needs to be embedded into the context of a reference coordinate system. GML 3.1 specifies a rich set of Earth bound coordinate systems, i.e.

```
<gml:Point gml:id="point96" srsName="epsg:4326">
        <gml:coordinates>31.5611 50.0041</gml:coordinates>
</gml:Point>
```

The example above uses the EPSG 4326 coordinate system. Earth bound coordinate systems are not applicable for other solar system planets. GML allows the specification of custom reference systems. *(howto define reference systems, i.e. custom names, mathematical ellipsoid, datums... Give example for the moon)*

Specifying a preset camera view is very useful for a web page that displays a particular focus on a geographic area. Multiple camera views can easily be defined, i.e. a camera with field of view (FOV) 45 degrees, looking at the location <wwml:lookAt> with north heading and 0 degree tilt:

```
<wwml:CameraFeature name="View1">
        <wwml:lookAt>
                <gml:Point srsName="epsg:4326">
                        <gml:coordinates>31.5611 50.0041</gml:coordinates>
                </gml:Point>
        </wwml:lookAt>
        <wwml:heading>0<wwml:heading>
        <wwml:distance>120</wwml:distance>
        <wwml:tilt>0</wwml:tilt>
        <wwml:fieldOfView> 45 <wwml:fieldOfView>
</wwml:CameraFeature>
```

Then, selecting a predefined camera view as the current focus is easily done by:

```
<wwml:currentView>
        <wwml:camera name="View1"/>
</wwml:currentView>
```

Geographic data content, feature collection

Finally, geographic data content is enclosed by a *gml:FeatureCollection* tag:

```
<gml:FeatureCollection>
        <gml:featureMember>

        </gml:featureMember>

        <!-- other feature members -->

</gml:FeatureCollection>
```

GML has a generic *gml:featureMember* tag. WWML subclasses this tag with *<wwml:PointFeature>, <wwml:LineFeature>, <wwml:PolygonAreaFeature>*, *<wwml:ImageOverlayFeature>* , *<wwml:LayerFeature>* and the previously introduced *<wwml:CameraFeature>*. The reason for this subclassing lie entirely within the internal mechanisms of the WorldWind XML parser, that is designed to readily identify the class of its feature.

```
<wwml:PointFeature>
        <wwml:drawPriority>100</wwml:drawPriority>

</wwml:PointFeature>

<wwml:LayerFeature>
        <wwml:drawPriority>600</wwml:drawPriority>

</wwml:LayerFeature>
```

XLink, dynamic features, remote sources

GML and therefore WWML explicitly supports the Xlink protocol. The normative xlink specification is available from W3C.

The Xlink functionality is best demonstrated in a simple example:

```
<gml:location>
        <gml:Point gml:id=”point96” srsName=”epsg:4326”>
        <gml:coordinates>131.453 52.4536</gml:coordinates>
        </gml:Point>
</gml:location>
```

is equivalent to

```
<gml:location xpoint:href="http://worldwind.nasa.gov/point96" xlink:title="another point"/>
```

Dynamic update protocol

This far, geographic features, even remotely linked ones are static. However linking to remote resources offers the opportunity to implement dynamic updating schemes. For example, temperature data remotely accessed is periodically updated every 10 minutes. WWML provides a framework to dynamically update WWML/GML specified features, that are remotely accessed. *Hard coded* values will be unaffected by the update mechanism!

A dynamic feature is defined by the attribute *dynamic* in the feature member tag. There are two update modes: *onlyInView* updates features only when they are in the current camera view, *always* updates features even if they are out of the current view.

A second important tag is the *updateProtocol*. The *automatic* value specifies that the data hosting server should negotiate a dynamic updating interval with the client browser. In the case of temperature data, the client will be informed that new data is available after 10 minutes. After the 10 minute interval, the client will issue a query to the server requesting new data. A second option is the *streaming* value. Here, the client signs up for a direct notification service with the server. Each time the monitored value changes on the server, the client will get notified about the changes and doesn't have to issue a separate request. A more detailed description of the client server update protocol can be found in a separate document.

```
<wwml:PointFeature dynamic="yes">
        <wwml:updateMode>onlyInView<wwml:updateMode>
        <wwml:updateProtocol>automatic</wwml:updateProtocol>

        <!-- feature description follows -->

</wwml:PointFeatures>

<wwml:AreaFeature>
        <!-- feature description follows -->
</wwml:AreaFeature>
```

Application Examples

The concepts of WWML/GML with semantic and syntax are best explained in application examples. In GML there is considerable freedom in specifying geographic content. At this time WorldWind doesn't support all the grammatical styles allowed in the GML specification. WWML defines best practices for coding and annotating geographic data content. To this extent WorldWind only supports the best practice methods specified in this WWML documentation.

Styles

Firstly, we define an overall screen display style that is used to show all the feature points labeled with the "City" class; This style has a fancy compound label showing the name & the size of the city in numbers, it also appears 10 meters above the actual surface and is accompanied by a default icon

```
<gml:FeatureStyle featureType="wwml:City">
        <wwml:DistanceAboveSurface> 10   </wwml:DistanceAboveSurface>
        <wwml:MinimumDisplayAltitude> 10000 </wwml:MinimumDisplayAltitude>
        <wwml:MaximumDisplayAltitude>600000</wwml:MaximumDisplayAltitude>
        <wwml:DefaultIcon xlink:href="file://C:\Documents and Setting\city.gif" />
        <gml:LabelStyle>
                <gml:style>font-family:Helvetia;font-size:20;font-style:bold;fill:green<gml:style>
                <gml:label>
                        City:    <gml:LabelExpression>//City/name</gml:LabelExpression>
                        , Size:  <gml:LabelExpression>//City/size<gml:LabelExpression>
                <gml:label>
        </gml:LabelStyle>
</gml:FeatureStyle>
```

The style encoded above will produce the following persistent label "City: XYZ, Size: 100,000" & a default icon. Following is another simple style, for a small town, no fancy compound labeling, no icons...

```
<gml:FeatureStyle featureType="wwml:SmallTown">
        <wwml:DistanceAboveSurface> 0 <\wwml:DistanceAboveSurface>
        <wwml:MinimumDisplayAltitude> 10000 </wwml:MinimumDisplayAltitude>
        <wwml:MaximumDisplayAltitude>600000</wwml:MaximumDisplayAltitude>
        <gml:LabelStyle>
                <gml:style>font-family:Helvetia;font-size:12;font-style:normal;fill:gray</gml:style>
                <gml:label>
                        <gml:LabelExpression>//SmallTown/name</gml:LabelExpression>
                </gml:label>
```

```
        </gml:LabelStyle>
</gml:FeatureStyle>
```

Style collections

Conveniently, the Xlink protocol can be used to reference the style definition from a remote location:

```
<gml:FeatureStyle featureType="wwml:City" xlink:href="http://worldwind.nasa.gov/City"/>
```

or similarly a collection of default styles can be remotely accessed by (I don't know if that is correct?)

```
<wwml:members xlink:href="http://worldwind.nasa.gov/defaultStyles"/>
```

Placemarks

Placemarks are the easiest & most important means of marking a location on Earth. Placemarks can have a rich set of attributes and interactions; they can be dynamic collection of random feature points (placemarks) with icons, names and additional data that is being displayed with persistent icon & labels, dynamic interactive descriptions when clicked on it & hyper links for further reference with specific display styles are given below:

In order to accomodate the various levels of attributes and interactions, several schemes are valid in WWML

Sparse list of richly annotated point features

In many cases, the user wants to insert only a few richly annotated placemarks, that can be interacted with on the graphical view. An easy way to accomplish this is by means of a *FeatureCollection* of *PointFeatures*:

```
<wwml:City>
```

```
<gml:FeatureCollection>
        <wwml:PointFeature gid="c1">
                <wwml:name>San Francisco</wwml:name>
                <wwml:size>700,000</wwml:size>
                <wwml:Icon xlink:href="file://C:/documents/sf.gif" />
                <wwml:description>
                        <![CDATA[Hello, you are in San Francisco!
                        Shown when cursor is over the location]]>
                </wwml:description>
                <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326" >
```

```
                    <gml:coordinates>138.6454,32.5454</gml:coordinates>
                </gml:Point>
        </wwml:PointFeature>
        <wwml:PointFeature gid="c2">
                <wwml:name>Oakland</wwml:name>
                <wwml:size>800,000</wwml:size>
                <gml:Point srsName="">
                    <gml:coordinates>138.6453,32.5454</gml:coordinates>
                </gml:Point>
        </wwml:PointFeature>
        <wwml:pointFeature xlink:href="http://worldwind.nasa.gov/anotherCity" />
</gml:FeatureCollection>


</wwml:City>
```

The example above also features another standard XML construct, the CDATA construct in *www:description* tag. The CDATA construct may encapsulates any character sequence, i.e. HTML, and prevents the content from being further processed by the XML parser.

Moderately long list of sparsely annotated point features

An alternative way to encode point feature attributes for a homogenous set of point features is via the "*domainSet*" and "*rangeSet*" coding. The *domainSet* is a linear list of point locations, each associated in order with a values in the *rangeSet*. In the example below, city locations & associated attributes are separated:

```
<wwml:City>
<gml:featureMember>
<gml:domainSet>
        <gml:MultiPoint srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:pointMembers>
                        <gml:Point gid="p1">
                                <gml:coordinates>132.645, 29.34</gml:coordinates>
                        </gml:Point>
                        <gml:Point gid="p3">
                                ...
                        </gml:Point>
                </gml:pointMembers>
        </gml:MultiPoint>
</gml:domainSet>

<gml:rangeSet>
        <gml:ValueArray>
                <gml:ValueComponents>
                        <wwml:name>San Francisco</wwml:name>
```

```
                    <wwml:name>Sausolito></wwml:name>
                    ...
                </gml:ValueComponents>
        </gml:ValueArray>
        <gml:ValueArray>
                <gml:ValueComponents>
                    <wwml:size>700000</wwml:size>
                    <wwml:size>50000</wwml:size>
                    ...
                </gml:ValueComponents>
        </gml:ValueArray>
        <gml:ValueArray>
                <gml:ValueComponents>
                    <wwml:temperature>52</wwml:temperature>
                    <wwml:temperature>49</wwml:temperature>
                    ...
                </gml:ValueComponents>
        </gml:ValueArray>
</gml:rangeSet>
</gml:featureMember>

</wwml:City>
```

In the example above, the *gml:domainSet* contains a point collection. Specifying point members can be done in equivalent ways, either with a single *gml:pointMembers* or multiple *gml:pointMember* :

```
<gml:pointMembers>
        <gml:Point gid="p1">
                <gml:coordinates>132.645, 29.34</gml:coordinates>
        </gml:Point>
        <gml:Point gid="p3">
                ...
        </gml:Point>
</gml:pointMembers>

<gml:pointMember>
        <gml:Point gid="p4">
                <gml:coordinates>134.645, 28.34</gml:coordinates>
        </gml:Point>
</gml:pointMember>
```

Even more useful is the remote xlink structure for querying the point positions:

```
<gml:domainSet xlink:href="http://www.somedata.org/citypositions.gml" >
```

Further, the example uses a *rangeSet* with aggregate values defined via *gml:ValueArray*. Instead of a value array, a data block representation *gml:DataBlock*, is more compact and is encouraged for medium length annotation files:

```
<gml:rangeSet>
        <gml:DataBlock>
                <gml:rangeParameters>
                <gml:CompositeValue>
                        <gml:valueComponents>
                                <wwml:name/>
                                <wwml:size/>
                                <wwml:temperature/>
                        </gml:valueComponents>
                </gml:CompositeValue>
                </gml:rangeParameters>
        <gml:tupleList>
                <![CDATA[San Francisco]]>,700000,52.3 <![CDATA[Sausolito]]>,50000,49
        </gml:tupleList>
        </gml:DataBlock>
</gml:rangeSet>
```

Long list of sparsely annotated point features

Though, the data block representation is an efficient way to encode larger chunks of numerical data, another way is to refer to the data in an external file. GML provides the file coding for defining a range set:

```
<gml:rangeSet>
        <gml:File>
                <gml:rangeParameters>
                <gml:CompositeValue>
                        <gml:valueComponents>
                                <wwml:name/>
                                <wwml:size/>
                                <wwml:temperature/>
                        </gml:valueComponents>
                </gml:CompositeValue>
                </gml:rangeParameters>
        <gml:fileName>http:/www.somedata.org/citydata.dat</gml:fileName>
        <gml:fileStructure>Record Interleaved</gml:fileStructure>
        <gml:compression>gzip</gml:compression>
        </gml:File>
</gml:rangeSet>
```

In particular the data file *citydata.dat* will be accessed remotely using *gzip* compression.

Dynamically updated features, interacting with views

WWML provides a framework to dynamically update WWML/GML specified features. Notice, that dynamic features only make sense in context with remotely accessed data. *Hard coded* position and values will be unaffected by the update mechanism.

```
<wwml:DynamicFeatures>
        <wwml:updateMode>onlyInView<wwml:updateMode>
        <wwml:updateNotification> <wwml:updateNotification>
        <wwml:pollUpdate> 10</wwml:pollUpdate>
        <!-- feature description follows -->
        <gml:featureMember> ... </gml:featureMember>
</wwml:DynamicFeatures>
```

Line features, line segments, graphs & vectors

Once again, for display purposes we start with a display style tag <wwml:MajorRiver>. Also, the <gml:featureMember> is only necessary, when a collection of features is considered.

```
<wwml:MajorRiver>
<gml:featureMember>
        <wwml:LineFeature>
        <gml:name>Sacramento River</gml:name>
        <gml:description>
                <![CDATA[Some HTML description of the Sacramento river]]>
        </gml:description>
        <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                <gml:posList dimension="2">
                0,50 70.12,30.12 100.0,24.34
                30.09,45,09  ..
                </gml:posList>
                ...
        </gml:LineString>
        <wwml:LineFeature>
</gml:featureMember>
</wwml:MajorRiver>
```

It shall be remarked that this example specifies the line segments via a list of coordinates encoded by *gml:posList*. This practice is encouraged instead of the <gml:coordinates> tag. The later one is still in use and is customizable, the symbols for decimal point **decimal**, component separation **cs** within tuples and tuple separation **ts** can be defined as:

```
<gml:coordinates decimal="." cs="," ts = " ">
                0,50 70.12,30.12 100.0,24.34 ..
</gml:coordinates>
```

The display style defined by <wwml:MajorRiver> can be imported from a separate location via XLink protocol. An example style definition will be presented below:

```
<gml:FeatureStyle featureType="wwml:MajorRiver">
        <wwml:DistanceAboveSurface> 0 <\wwml:DistanceAboveSurface>
        <wwml:MinimumDisplayAltitude> 0 </wwml:MinimumDisplayAltitude>
        <wwml:MaximumDisplayAltitude>600000</wwml:MaximumDisplayAltitude>
        <gml:LabelStyle>
                <gml:style>
                        font-family:Helvetia;font-size:12;
                        font-style:normal;fill:blue
                </gml:style>
                <gml:label>
                        <gml:LabelExpression>//MajorRiver/name</gml:LabelExpression>
```

```
                </gml:label>
        </gml:LabelStyle>
        <gml:GeometryStyle geometryType=”gml:LineString”>
                <gml:style>
                        stroke:blue;stroke-width:”4px”;stroke-opacity:0.8;
                </gml:style>
        </gml:GeometryStyle>
</gml:FeatureStyle>
```

The <gml:style> tag elements are closely modeled after corresponding SVG line stroke attributes:

```
<gml:style>
        stroke:white;              // color parameter for stroke
        stroke-width:2px;          // 2 pixels wide stroke
        stroke-opacity:.6;         // opacity of 0.6
        stroke-dasharray:5,3,2; // controls the pattern of strokes, a list of length parameters
</gml:style>
```

Area features, polygonal objects

Similarly to a collection of point and line features is the collection of polygon (area) features. The user has several ways to define and annotate area features. The following description is suitable for a small number of polygonal features, each richly annotated:

```
<gml:featureMember>
        <wwml:AreaFeature>
                <gml:name>Lake Pino Grigio</gml:name>
                <gml:description>
                        <![CDATA[This description will show up a HTML page when cursor is
                                moved over the lake]]>
                </gml:description>
                <wwml:area>5343.43</wwml:area>
                <wwml:avg_depth>154.53</wwml:avg_depth>
                ... <!-- some other area descriptors -->
                <gml:boundedBy>
                        <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                                <gml:posList>
                                        -132.16563,57.1034  -131.24353,57.98575
                                <gml:posList>
                        <gml:Box>
                </gml:boundedBy>
                <wwml:extent>
                        <gml:Polygon gid="poly1" srsName="epgs#4362">
                        <gml:exterior>
                                <gml:LinearRing>
                                        <gml:posList> 0,0 10.6,1.0 5.43,4.2
                                                ...
                                        </gml:posList>
                                </gml:LinearRing>
                        </gml:exterior>
                        <gml:interior>
                                ...
                        </gml:interior>
                        </gml:Polygon>
                </wwml:extent>
        </wwml:AreaFeature>
</gml:featureMember>
```

The core sequence that defines a polygon area is enclosed by *gml:Polygon.* The polygon is defined by closed line rings. As usual in 2 dimensions, polygon areas can have holes which are identified by the one or several *gml:interior* tags. The *gml:exterior* tag indicates the outer polygon boundary, several of those tags can be used for non-contiguous polygons.

Moderately long list of sparsely annotated area features

Legacy Data Formats (Shapefiles)

Shapefiles are a legacy standard for line, point and area features in the GIS world, see [2]. The ESRI shapefile data format efficiently stores features and annotations in binary files. The underlying binary format requires special software for building and modifying shapefiles.

Further, the technical shapefile documentation does not contain any display styling options nor references to metadata, i.e. the author, data of origination, publisher, keywords.

GML does not support shapefiles! Special translators have to be written that would translate shapefile content into lengthy GML coded content. Thus computational performance while loading and streaming GML coded data could be severely impacted.

The WorldWind markup language as an extension to GML has direct support for ESRI shapefiles:

```
<wwml:shapefile xlink:href="file://Users/blah/rivers.shp">
```

Overlay Imagery

```
<wwml:ImageOverlayFeature>
        <wwml:name>My town from a birds view</wwml:name>
        <wwml:description>
                The city of fantastic ice cream everywhere
        </wwml:description>
        <wwml:ImageData xlink:href="file://Users/blah/mytown.png" />
        <gml:RectifiedGridCoverage>
                <gml:boundedBy>
                        <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
                                <gml:coordinates>
                                        -132.16563,57.1034  -131.24353,57.98575
                                <gml:coordinates>
                        <gml:Box>
                </gml:boundedBy>
                <gml:TimeStamp>
                        <gml:TimeInstant>
                        <gml:timePosition>2005-01-17T00:00:00:+09:00</gml:timePosition>
                        </gml:TimeInstant>
                </gml:TimeStamp>
        </gml:RectifiedGridCoverage>
<wwml:ImageOverlayFeature>
```