ARML - Augmented Reality Markup Language

Martin Lechner
CTO, Mobilizy GmbH.
Ginzkeyplatz 11
5020 Salzburg, Austria
martin.lechner@mobilizy.com

October 5th, 2010

Abstract
This paper was designed for the International AR Standards Workshop, held on October 11th and 12th, 2010. The paper covers ARML (Augmented Reality Markup Language) in the existing format (v1.0) and in a proposed successor-format (v2.0).

1 ARML v1.0

ARML v1.0 was proposed by Mobilizy, the company behind the Wikitude World Browser, in the course of the release of Wikitude 4 in January 2010. ARML v1.0 is based mainly on KML, with the following main differences between ARML and KML:

- KML is a very rich standard. Most of the tags provided are not (yet) necessary for AR applications. ARML reduces the KML standard to the bare minimum.
- ARML adds some additional features to the KML standard, see section 1.1.3 for details.

1.1 ARML v1.0 structure

An ARML file consists of two sections. One section provides information about the Content Providers, which are the provider of sets of POIs (e.g. Wikipedia). The second section defines the POIs itself.

1.1.1 An empty ARML Document

A minimal ARML document comprises the following tags:
The namespaces represent the “acceptance-status” of the tags in the namespace.

The kml namespace is the native KML namespace. The ar namespace should include all tags which have been found to be relevant for all (or most of the) AR browsers. The wikitude-namespace defines the proposed tags which Wikitude uses and Mobilizy believes will be relevant to all AR browsers. Other AR browsers might propose additional namespaces, tags within these namespaces should constantly be reviewed and, if found in various browsers, should then be merged into one tag in the ar namespace. Finally, one could imagine to move the ar-namespace back to the KML standard to have the KML standard enriched with AR data in the future.

Figure 1: ARML Namespace structure

1.1.2 The Content Provider Section

The content provider section holds all information about a single Content Provider, like the name, a description, a logo, the web page of the provider, etc. The provider shares this information with all the POIs which it delivers:

<ar:provider id="myCpId">
  <ar:name>The name of the CP</ar:name>
  <ar:description>
    Any description of the CP
  </ar:description>
  <wikitude:providerUrl>
    http://myUrl.com/
  </wikitude:providerUrl>
</ar:provider>
1.1.3 The POIs Section

Every POI needs to be linked to one (and only one) Content Provider stated above. It comprises the essential AR-data (location, name, description, phone number, email etc.) for each POI.

```xml
<Placemark id="123456">
  <ar:provider>myCpId</ar:provider>
  <name>Title of my POI</name>
  <description>My POI description</description>
  <wikitude:info>
    <wikitude:thumbnail>http://thumbnailUrl.com</wikitude:thumbnail>
    <wikitude:phone>123-456-789</wikitude:phone>
    <wikitude:url>http://poiUrl.com</wikitude:url>
    <wikitude:email>inf@myPoi.com</wikitude:email>
    <wikitude:address>
      My POI Street 5, 5020 POI, Austria
    </wikitude:address>
    <wikitude:attachment>
      http://myAttachmentLink.com
    </wikitude:attachment>
  </wikitude:info>
  <Point>
    <coordinates>
      13.048056,47.797222,432.0
    </coordinates>
  </Point>
</Placemark>
```

The coordinates of the POI need to be provided as longitude, latitude and altitude. Altitude is optional.

2 ARML v2.0

ARML v2.0 will be the successor of ARML v1.0. ARML v2.0 is still in the “idea-phase”, thus only core ideas and high level improvements will be presented in this paper (without going into deep details):
1. Action Scripts (ARAS)

2. Reuse existing KML tags

3. Design Customization Options

4. Navigation Support

5. 3D model support

6. Linked Data support

Each idea will be discussed in a separate section.

2.1 Action Scripts (ARAS)

For a great AR experience, it is absolutely essential that users can interact with the browser, for example fire actions according to well-defined action triggers. Just as JavaScript is enriching web browsers, an Augmented Reality Action Script (ARAS) should enrich Augmented Reality Browsers. One might think of location triggers, geo fences, pressing certain areas (like buttons) in an AR Browser etc. Actions might include, but are certainly not limited to showing content, executing certain pieces of code etc.

ARAS should be designed to be as open and extensible as possible. As a first step, the Action Script should allow action triggers and actions which are currently common throughout the AR browsers.

To get an idea of what an AR Action Script might look like, and to start a discussion, here is a first draft of the Action Script, as Mobilizy has it in mind:

```xml
<aras>
  <triggers>
    <trigger id="myTrigger">
      <priority>1</priority>
      <fireOn>
        <proximity>
          <latitude>47.21</latitude>
          <longitude>12.29</longitude>
          <radius>50</radius>
          <unit>m</unit>
          <proximity>
            <fireOn>
              <actionFired>myAction</actionFired>
            </fireOn>
          </proximity>
        </fireOn>
      </proximity>
    </trigger>
    ...
  </triggers>
  <actions>
    <action id="myAction">
```
ARAS is entirely following XML syntax and should be specified within the ARML file. Each trigger is defined in its own tag, stating the priority of the trigger (in case multiple triggers would fire at the same time), the event the trigger will react on, and the action that will be fired.

Actions are defined analogous. The actual action which should be executed is specified in JavaScript. The AR Browser will provide objects which can be modified (markers, balloons, POIs) and actions which can be executed directly (taking a photo, starting another application etc.) through JavaScript.

2.2 Reuse existing KML tags

When comparing ARML v1.0 to the existing KML standard, we see that they diverge in areas where it’s not absolutely necessary. One example would be the presentation of a POI marker. KML allows for customization of each marker on the POI level as well as bundling the style of multiple markers through the style-tag, whereas ARML v1.0 allows customization on Content Provider level only. It is be possible to reuse the KML style-tag to achieve customization on Content Provider level and on POI level. Other examples would be the address tag on POI level, various altitude modes etc.

To conclude, ARML shall reuse existing ideas and structures from KML.

2.3 Design Customization Options

ARML v1.0 only allows the customization of the actual data, but not how this data should be rendered in an AR browser. ARML v2.0 should include support for design options as well. HTML might be the best option to also define how the data should be presented.

2.4 Navigation Support

ARML should also allow turn-by-turn AR Navigation. The data must include Way and Routing Points (latitude, longitude and altitude), as well as some additional information for the routing points, like the type of the turn etc.). Also, some additional things like the road design etc. should be customizable.

2.5 3D model support

KML supports viewing of 3D models. While some tags defined in KML for 3D models might not be necessary for AR (like LookAt), KML already provides
the tags to specify 3D models. ARML should make use of these tags to be able to handle 3D models.

2.6 Linked Data support

According to the ideas of Semantic Web, future data will be available in the Linked Data Cloud. ARML v2.0 should already provide the possibility to specify data in RDF format.

3 More information

More information on ARML can be found on the ARML webpage (see [2]).

References