XML BASED DATA DESCRIPTION FOR THE PHOTOGRAMMETRIC DOCUMENTATION OF HISTORIC BUILDINGS

Guenter Pomaska
Bielefeld University of Applied Sciences, Faculty of Architecture and Civil Engineering, Germany
gp@imagefact.de - www.imagefact.de

KEY WORDS: 3D, Archiving, CAD, Close Range, Internet/Web

ABSTRACT:

XML Extensible Markup Language and related technologies establish the basics of second-generation Web applications. The First-generation Web was based on HTML with respect to the design of publication. Today we focus on structured storage of information, apart from the application.

The documentation of the Citadel in Wesel stands as an example for a photogrammetric application. The complete photogrammetric database, including control points, camera data, orientation data and detailed graphical evaluation, is carefully worded as a XML file format. A XSL processor provides a transformation into HTML for displaying numerical values in a human readable form.

Goal of this study is to provide registered images for further evaluation on distributed systems. A map in SVG displays a symbol for each camera position. This symbol acts like a container, including all the necessary data for further evaluation. Activating the link, related to that icon, starts a PHP-Script at the server site. The script initiates a XML parser and generates a Web page, making available interior and exterior orientation as well as image data at the client site. Applying the client's JavaScript utility enables measurements upon request.

Scalable Vector Graphics is the standard for displaying 2D-graphics on the Web. SVG as an XML application, prepares graphic for interactivity and animation. Applying style sheets provides an increasing number of design options. Remaining is the 3D-graphic structure. We use the DTD for extensible 3D, the XML formulation of the former VRML format.

Translation utilities, viewer and software concepts for the exemplified duties in a photogrammetric application, applied to building documentation, are introduced in this article. One can follow the concept, visiting http://www.imagefact.de/zitadelle-wesel.

1. STATIC AND DYNAMIC WEB SITES

1.1 Client-Server Architecture

Applying a client-server architecture does not require a computer network configuration. One can install a Web server and a Web browser on the same computer. The client (here the browser) sends a request to the server, addressed by *localhost*. The server is looking for the requested document, transfers the document to the client and the browser is rendering the file information into a readable format. The language, a browser understands, is HTML Hypertext Mark-up Language. Communication between server and client is agreed via a protocol, the Hypertext Transfer Protocol. The process is known under the term static Web application. Major software components of the Web are the Web browser, the HTTP protocol, the description language HTML and the Web server.

1.2 Client Sided Dynamic

While a Web site is changing its appearance or content by user interaction without loading a new document, we are talking about client sided dynamic. The document object model DOM enables access to any object (element) and its attributes, that is included in a Web document. The DOM is implemented in JavaScript, an extension to Web browsers. We focus here to a small application calculating control points by spatial intersection.

JavaScript has a hierarchical structure of application objects. Under the document level there are form objects existing. A form itself, is followed by elements. Forms and elements can be identified by names. If we want to set the value of one of the input fields in the sample shown in figure 1, we have to code *document.vws.xctrlpt.value = "100.00"*; *vws* stands for the name of the form, *xctrlpt* is the name of an input field, value is the attribut for the content.

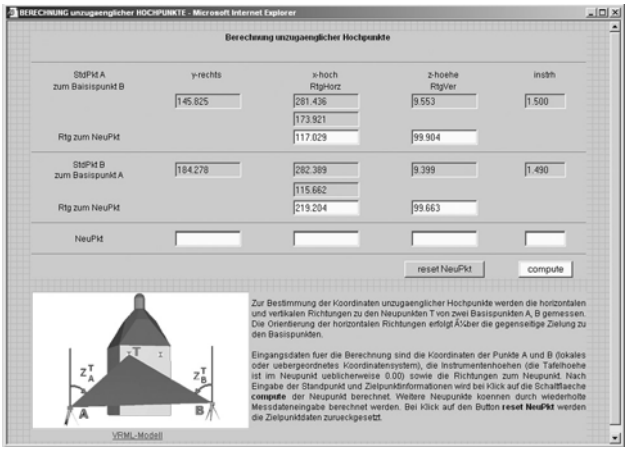

Figure 1. Calculating control points applying JavaScript

The HTML coding, applying the input-tag, reads as follows:

```
<form  name="vws">
  <input name="xctrlpt" value=" "
         type ="text" >
```

```
  </input>
</form>
```

Inside the angle brackets more attribute definitions can occur. This short explanation will help understanding the data exchange between a Web site and PHP programs.

1.3 Server Sided Dynamic Applying PHP

PHP Hypertext Pre-Processor is a server extension. The language is embedded in HTML documents. If the server detects PHP commands, the code will be carried forward to the PHP interpreter where the HTML code will be first generated and after processing submitted to the client. The PHP code is not visible for the client. The browser receives the resulting HTML code only. The PHP interpreter supports amongst others XML processing, SQL data base functions, graphics and file access.

2. DESIGN OF A XML STRUCTURE

2.1 XML Extensible Mark-up Language

XML Extensible Mark-up Language is a meta language to structure information. XML uses tags, keywords in angle brackets with additional attributes, to enclose content. Compared with HTML, the denotation of the tags is not defined and will be interpreted afterwards by the application.

XML files are plain text files. Rendering XML files requires other technologies. XML documents are starting with a prologue, followed by the root element. The prologue displays details for the (DTD Document Type Definition) as stated below. The close-range photogrammetric application will demonstrate the principle.

```
<?xml version="1.0" encoding= "ISO-8859-1"
standalone="yes" ?>
<!DOCTYPE        imageBundle        SYSTEM
"imageBundle.dtd">
<imageBundle>
     <!--Inhalt des Dokuments -->
</imageBundle>
```

The root element *imageBundel* instances a document class, that is defined in the !DOCTYPE statement. An element and its subelements will be represented in the tree structure by nodes. The element itself consists of a start-tag, the content and an end-tag. Attributes will added into the start-tag. A photogrammetric camera definition may look as follows:

```
<cameraData>
  <camera>
    <type>nikon_28</type>
    <ck>-18.23718</ck>
    <xh>-0.09973</xh>
    <yh>-0.01304</yh>
    <a1>-3.03846E-004</a1>
    <a2>6.43569E-007</a2>
    <formX>23.462</formX>
    <formY>15.600</formY>
  </camera>
</cameraData>
```

A Web browser displays the tree structure of that document. By simply clicking the symbols, the nodes can be closed or opened, as shown in figure 2.

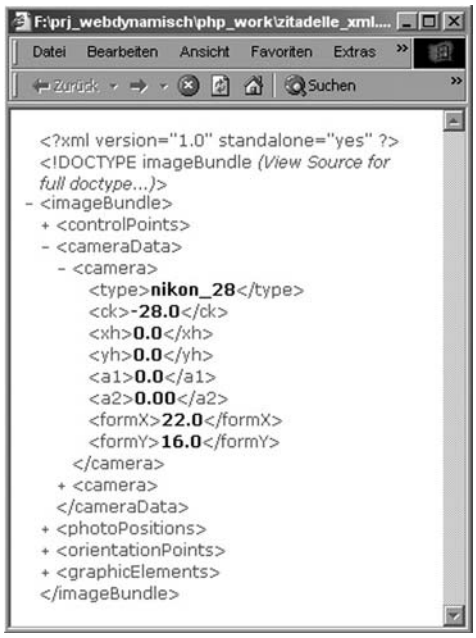

Figure 2. Tree structure of a XML document

2.2 XSL Extensible Stylesheet Language

The XSL Extensible Style Sheet Language provides transformation from XML into another, better readable format. A XSL processor can be applied by the Web browser or, as an alternative. server sided by the application. An off-line transformation results in a HTML document to be stored on the server. Another important tool for transformations is XPATH. XPATH provides search patterns and enables extractions form XML documents. An external XSL file must be referenced in the prologue using the statement:

```
<?xml-stylesheet version="1.0"
href="template.xsl" type="text/ xsl" ?>
```

We do not discuss the details of XSL here. An application of the for-each and value-of select statement may be give an idea how a XSL document is parsed by a XSL processor:

```
<xsl:for-each
   select="the search pattern">
   <tr>
    <td class = "tab_value"
    <xsl:value-of select="type"/>
    </td>
  </tr>
</xsl:for-each>
```

HTML tags must be combined with XSL statements. The prefix xsl: defines the namespace, the class definition defines the appearance of the element in the browser.

2.3 DTD Document Type Definition

XML documents can be well formed or guilty. A well formed document becomes a guilty document by adding a DTD. A DTD defines the grammar for information processing of the XML file. All elements, attributes entities and specifications about quantity, content and nesting of elements must be predefined in a DTD. An extract of the DTD, applied in the example reads as follows:

```
<!DOCTYPE imageBundle [
  <!ELEMENT imageBundle (controlPoints*,
            cameraData* ,
            photoPositions*,
            orientationPoints*,
            graphicElements*)>
     <!ELEMENT controlPoints (point*)>
        <!ELEMENT point (pnr*, code*, x+, y+, z+)>
             <!ELEMENT pnr     (#PCDATA)>
             ...
     <!ELEMENT cameraData (camera+)>
        <!ELEMENT camera
         (type+, ck+, xh+, yh+, a1+, a2+,
          formX+, formY+)>
               <!ELEMENT type (#PCDATA)>
               <!ELEMENT ck   (#PCDATA)>
               <!ELEMENT xh   (#PCDATA)>
               <!ELEMENT yh   (#PCDATA)>
               <!ELEMENT a1   (#PCDATA)>
               <!ELEMENT a2   (#PCDATA)>
               <!ELEMENT formX(#PCDATA)>
               <!ELEMENT formY(#PCDATA)
```

Details can not be discussed here. It must be annotated, that the DTD will be replaced in future by the more powerful Xschema.

With XML, XSL and DTD structured information is separated from processing: This file structure can be processed with object-oriented programming languages in client-server environments.

3. PARSING XML FILES WITH PHP

3.1 The Parser

It is proposed to explain parsing and processing XML-documents by using the photogrammetric example from above. Requested are the camera values for the camera referenced by the description *nikon_28*. The client's request typed in the address line of the browser:

```
http://www.imagefact.de/zitadelle-wesel/
 parse_camera.php?camera=nikon_28
```

has to be submitted to the server.

A parser will go through the file until the requested camera is found. The parser analyses and validates the file structure. Expat is an event driven parser and provided by PHP. Events are the occurrences of tags and content. An instance of the parser is created with the statement:

```
$parser = xml_parser_create();
```

It is necessary to set parameter for the parser and handler for the elements, for example:

```
xml_set_element_handler( $parser,
   "start_element", "end_element" );
or
xml_set_character_data_handler(
    $parser, "inhalt" );
```

start_element , *end_elemen*t and *inhal*t are functions called by the parser, if it meets one of the defined events.

3.2 Processing Data

The function *inhalt* will process, if the content of a tag was *camera*. The camera data is stored in arrays.

```
function inhalt($parser,$data ){
 global $curr_tag,$index,$camera
 switch ($curr_tag){
  case "type": $index=$data;break;
  case "ck": $camera[$index][ck]=$data;break;
  case "xh": $camera[$index][xh]=$data;break;
  case "yh": $camera[$index][yh]=$data;break;
  case "a1": $camera[$index][a1]=$data;break;
  case "a2": $camera[$index][a2]=$data;break;
  case "formX": $camera[$index][formX]=
                $data;break;
  case "formY": $camera[$index][formY]=
                $data;break;
  }
}
```

After getting the camera data, the arrays must be evaluated and the requested data, embedded in HTML-tags, has to be generated and must be passed to the client. PHP uses the echo-function for writing HTML commands. Figure 3 shows the answer to a request including image data as rendered by the browser. The complete examples can be found on the Web under *www.programmierpraktikum.de*, follow the navigation to the readers section (leserbereich), select PHP&XML there from the menu.

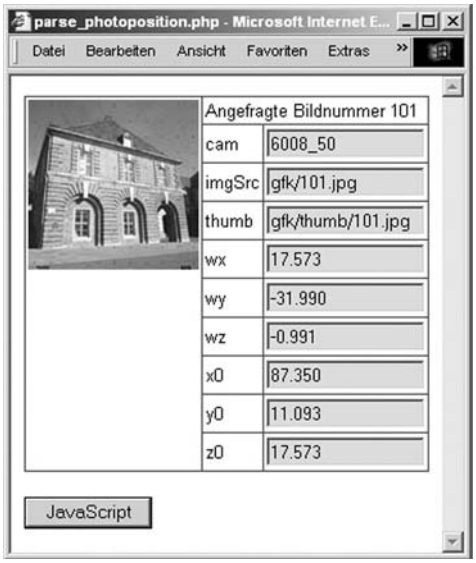

Figure 3. Parsing XML-Information and displaying the result on a Web page

4. PROCESSING GRAPHIC DATA

4.1 SVG Scalable Vector Graphics

Scalable Vector Graphics SVG is the XML formulation of 2D vector graphics. It includes drawing of vector data, displaying of image data, interaction and animation. Structure and appearance of graphic elements is separated by using style sheets

Applying a SVG viewer as a plug-in for the Web browser enables zooming and panning in the graphic area. Figure 4 displays the position of control points and demonstrates the use of image data inside vector data.

Figure 4. Image data nd vector data combined with SVG

The images are rectified and represent to a certain extend a true scale image map. Zooming in is followed by loss of quality. This is not the case with vector data. The appearance of the drawing, shown in figure 5, is dependent from the applied style sheet. The file content can be easily transformed to different media, like large format printers or PDAs. Furthermore sophisticated design possibilities like pattern filling, shading, insertion of symbols and others are provided by SVG.

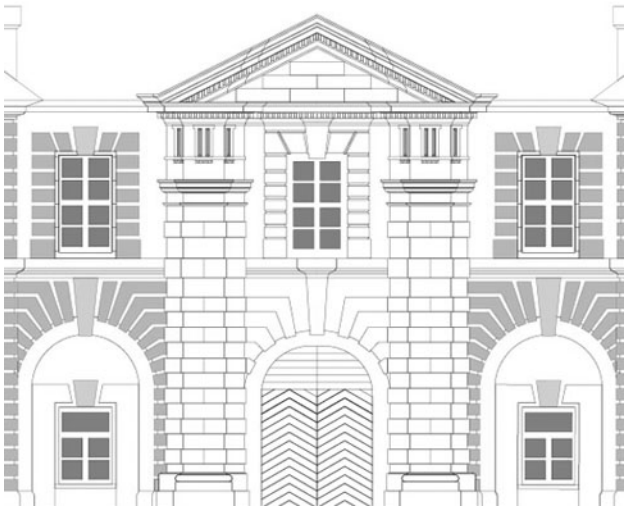

Figure 5. Appearance of a SVG file is driven by a style sheet

SVG enables among other features interactivity and animation. The photogrammetric application benefits from interactivity. Registered camera positions are marked in the xy-drawing using symbols. Those symbols carry a link to an evaluation software, that has access to the image data. Clicking a camera symbol loads a page, enabling the user to take on-line measurements. A first version of this software, written in PHP, is implemented with single image measurement functions. The PHP interpreter allows access to the file system of the server. From there measurements can be stored in the XML structure. Figure 3 is part of that evaluation software. The image coordinate measurement is client sided performed by means of JavaScript.

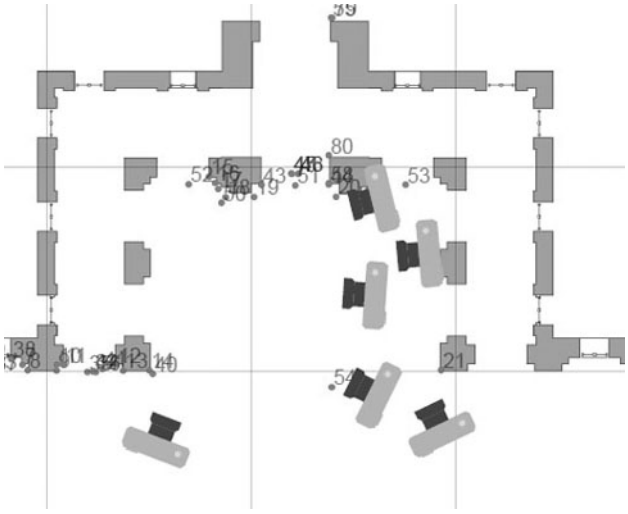

Figure 6. XY-Drawing of the exposure arrangement

4.2 X3D Extensible 3D

3D graphics data for the Web is known as VRML Virtual Reality Modeling Language description. That format convention is updated to the XML version, predicted as X3D. The OCTAGA viewer can be applied for VRML and X3D in stand-alone mode or as a plug-in for Web browsers. Virtual 3D models are coded in such a scene graph format. One can navigate through the building inside the Web browser or download the sometimes very huge files in advance. Visiting the project's Web site will guide the user to a virtual reconstruction of the roof.

5. CONCLUSION

XML as a meta language is designed for structuring information and separating information from processing. The family of XML-languages and tools, like XSL, XPATH, SVG or X3D, provide device and platform independent processing. XML structured information can be prepared for displaying on monitors and PDAs or printing. The above given examples, taken from a photogrammetric project, stay for a variety of application potentials. The sketched code here is printed to give an impression of the need for teaching and learning XML based Web technology.

References from Books:
Pomaska, G., 2005. *Grundkurs Web-Programmierung*. Vieweg-Verlag, Wiesbaden.

References from Other Literature:
Pomaska, G., 2003. *Introduction of SVG as a data interchange format for architectural documentations.* CIPA International Symposium, Antalya, Turkey.

Pomaska, G., Dementiev, N. , 2005, *XML basierte Datenformulierung zur Web-konformen Dokumentation photogrammetrischer Bauaufnahmen.* PFG Zeitschrift für Photogrammetrie, Fernerkundung und GeoInformation

References from Web sites:
http://www.programmierpraktikum.de
http://www.imagefact.de/zitadelle-wesel
http://www.adobe.com/svg
http://www.octaga.com