Evaluation and visualization of 3D models using COLLADA parser and WebGL technology

MIROSLAV MILIVOJEVIĆ, IGOR ANTOLOVIĆ, DEJAN RANČIĆ
Computer Graphics and Geographic Information System Laboratory
Faculty of Electronic Engineering, University of Niš
Aleksandra Medvedeva 14, 18000 Niš
SERBIA
miroslav.milivojevic@elfak.ni.ac.rs, http://www.elfak.ni.ac.rs

Abstract - This paper describes the evaluation of the developed importer, as a subset of a larger GinisVisAjaxLibrary. The basic functionality of the importer is to read files in the COLLADA format, which is one of the most popular format today for the representation of 3D models. A lot of advantages derive from the architecture of this format, and the most popular are: handling of complex data over the Internet, the completeness of information needed for the visualization of complex 3D scenes, structural arrangement of data as well as, extensibility of the format by using extra tags. The importer was evaluated by determining the performance results presented in the form of tables and charts that include: downloading time, parsing time, number of textures, number of nodes, number of polygons, FPS and, FPS in an empty scene. The tests were executed in well-known browsers and on different operating systems.

Key-Words: evaluation, WebGL library, COLLADA format, 3D models, importer, visualization

1 Introduction
During the previous couple of years, 3D data representation, as a substitute for desktop environments for 3D visualisation [1,2], presents an important factor on the Web. In many domains of modern 3D implementing applications, methods for presenting and managing sets and subsets of data have become more frequent, and this is exemplified by Web application which adequately represent: complexes of buildings, museum interiors, flats, kitchens, chemical compounds, human body (Body Browser [3]), as well as a large amount of georeferenced data (GIS applications [4]).

The implementation of this kind of Web applications requires standard or specialized modular Web services [5], databases, engines, environments, libraries and formats in order to represent complex elements on the scene. Standardized formats which include all the required elements for representing 3D data on the scene, inserting animations and physics into the scene, as well as model description models, usually have a high level of data transferability, compatibility and usefulness. There are a large number of formats of this kind, but the most used ones include: VRML, X3D, COLLADA[6], OBJ. COLLADA is a free open standard which uses an XML scheme to present 3D content. This format has been developed along with X3D standard in the last couple of years.

Although these two standards are very similar regarding 3D data structure representation technology, they differ in their aims and purposes.

This paper describes an evaluation of an importer that we developed, which parses and converts 3D contents written in the COLLADA format into an universal mesh structure. Parser and mesh are structures which are a part of a larger Ajax based 3D data visualisation library called GinisVisAjaxLibrary [7]. GinisVisAjax Library uses WebGL [8] technology in order to visualise 3D data in Web browsers. The paper then describes the structure of the COLLADA format, its standardized purpose, as well as environments and applications which use it. The specially designed architecture of the parser enables universality and uniformity of different objects encompassed in the mentioned COLLADA format. Results are presented in tables and graphs, and were obtained in the process of testing in the most frequently used Web browsers and operating systems.

2 COLLADA file structure
COLLADA is an abbreviated name of COLLABorative Design Activity. This format defines an XML scheme which enables 3D data presentation, simple editing of digital data and compatibility with many 3D tools. This format
facilitates the interaction of applications over data in 3D space. COLLADA is designed by an unprofitable technological consortium Khronos Group [6].

During the first phase of COLLADA scheme design, it was necessary to establish following assumptions: This is not a format that will be used by a game-engine. This format should be used for tools design and 3D content that are supposed to provide the usage of modern interactive application concepts. Many applications use COLLADA format, but not as a final mechanism for 3D data delivery. Final users develop and test relatively simple 3D content rather quickly. Model testing includes the usage of advanced rendering techniques, such as vertex and pixel programs (shaders[9]). These assumptions led to setting the goals which COLLADA format has managed to achieve; some of the goals[6] are:

1. To convert independent digital assets from proprietary binary formats into a well-specified, XML-based, open-source format.
2. To provide a standard common format so that COLLADA assets can be used directly in existing content tool-chains, and to facilitate this integration.
3. To provide an easy integration mechanism that enables all the data to be available through COLLADA.
4. To be a basis for common data exchange among 3D applications.
5. To be a catalyst for digital-asset schema design among developers and digital content creation (DCC), hardware, and middleware vendors.
6. To be adopted by as many digital-content users as possible.

A large number of advantages of this format is derived from the XML technology. XML provides a well-defined framework. Character sets such as: ASCII, Unicode, Shift-JIS are already covered by the XML standard, so when end users make some schema of data using the XML technology it impacts on increasing interoperability. The example of XML schema allows for the technology to be fairly easy understandable without using its documentation. Another advantage is the existence of XML parser in almost every programming language for any platform. It makes files easily accessible for any application. Using this, developers can create very powerful tools and this is made possible by the combination of multiple software packages and using of known technology today. COLLADA format allows transport of data from one application for digital content creation to another. There are a lot of commercial applications that support the use of this format and they are as follows: Maya (ColladaMaya), 3ds Max (ColladaMax), Poser (v.7.0), LightWave 3D (v.9.5), Cinema 4D (MAXON). The most popular open-source applications and libraries that support this format are: Google SketchUp, Blender and GLGE.

COLLADA is focused on creating a series of commercial and freeware tools. These tools allow the transmission and display of 3D content, especially in applications implemented from fields of science such as medicine, astronomy, chemistry and so on. This paper describes the evaluation of the COLLADA format. COLLADA format is one of the intermediate format (medium format type). The primary goal of this format is the presentation of complex 3D models in a way that enables communication between tools and applications as well as adapting to different platforms. Extra tags allow the definition of user information such as descriptions of the models. COLLADA documents that describe digital assets are XML files, usually identified with a .dae filename extension that means a digital asset exchange. The same method can be used for transfer or processing of 3D content tools of a scientific software solutions. Advantages and usage of this format is great, as follows: suitable for delivery of interactive 3D Web content and business applications that require the transfer of 3D data in real time, COLLADA format is very compatible with Google Earth and supported in many Web applications. This feature applies at the following domain of applications: online advertising, computer games, virtual worlds (virtual reality), social networks, GIS (Geographic information system) applications, medical visualization etc.

The following text explains the important elements of the COLLADA format. These elements are represented by XML tags or referenced to another element which are an integral part of them and some of them are: library visual scenes, library materials, library effects, library geometries and library images.

<library_visual_scenes>
  <visual_scene id="vis_scene"/>
</library_visual_scenes>

<scene>
  <instance_visual_scene url="#vis_scene"/>
</scene>

Here is an example of two elements such as library_visual_scenes and scene element. The scene element consists of an instance_visual_scene element which references to a property visual scene element. There are cases where there are more scenes and their definitions are covered in
library_visual_scenes element. The scene may contain various elements that describe the different ambients of models, models and the different angles of viewpoint.

```xml
<library_geometries>
  <geometry name="cube" id="cube123">
    <mesh>
      <source id="box-Pos"/>
      <vertices id="box-Vtx">
        <input semantic="POSITION" source="#box-Pos"/>
      </vertices>
    </mesh>
  </geometry>
</library_geometries>
```

A material can have both image and colors. There are a lot of materials and images that must be loaded for one model and it could affect performance.

**Library_images** is an element that consists of a path or relatively path to the images. Image has a image name and, a path that is embedded into int_form element. Image name must be matched with a text that is embedded into newparam which is embedded into effect.

### 3 Parser designing

Development process of the parser provides its main function such as a readability of models for some library, tool or framework. This is a process of translation from one model to another. In this case it is necessary to translate the model from COLLADA format to a mesh.

Mesh is a universal structure that is used in a larger Ajax library[7]. This library manages with the WebGL API and uses information from the mesh for model visualization. Figure 1 shows the architecture of the universal parser. Strategy design pattern provides a selection of the right parser. The instance of Mesh Manager classes initiates data parsing from the COLLADA document and uses the parsing results for a mesh loading. Load method uses GET or POST request in the URL for the model downloading.

![Fig.1 The architecture strategies for choice of parser](image)

**MeshModel** class is derived from the **Mesh** class. Instance of this class creates **MeshManager** object in the method parse, then calls its parsing method. Render method has a canvas as an argument that contains a reference to the renderer and the HTML canvas element.

**Mesh** class is composed of geometry and materials. Render method has a property that initiates that the mesh is visible or not. Mesh is drawn in the Canvas class if it is visible. The architecture of the COLLADA parser is composed of several classes such as: LibraryScenes,

XmlDocument class is the basic XML parser. This parser consists of methods to search and return value of the attribute node, then methods for searching nodes and return of the node based on their tag name, as well as methods for searching nodes and return of the node based on attribute name or based on attribute values.

Manager class initiates the main parse method. This method runs load method of the Libraries class. Manager class initiates the main parse method. This method runs load method of the Libraries class. Then same method runs loadMesh method, and Manager object initiates logic to manage Libraries class.

Fig.2 The architecture of the COLLADA parser

This allows the parsing of the individual parts of the COLLADA format that is very important to create information that is necessary for the scene.

Mesh object is created using the method createMeshInstance when the following conditions are met: when the required data is extracted and when the information are obtained that fully describes the 3D scene and models.

Libraries class contains all the references for the class libraries such as scene, material and geometry. So, when the required Libraries class call methods of objects of these classes and reference sequence is defined by the logic of parsing in class Manager.

LibrariesScenes class contains all the scenes covered in the COLLADA format, so this is a class which instance initiates the start of parsing. Each instance of the scene contains a reference to a scene that is defined in the class LibraryVisualScenes.

LibraryVisualScenes is a class that contains all the defined scenes in a COLLADA document. The scenes contain nodes or node references that are defined in the LibraryNodes class. This class also contains the cameras or camera references.

LibraryImages class contains full or relative path to images.getImage method returns a requested path using the required id attribute.

LibraryNodes class contains nodes that contain the necessary elements, such as references to geometry, references to materials and transformation matrix or references to nodes covered by this class. In this way, grouping nodes and creation of a tree of nodes is achieved. This mesh can have multiple sub-mesh structures or contains only the previously mentioned elements (leaf tree).

LibraryGeometries class contains geometry instances that are referenced in the nodes in the LibraryNodes class. Geometry described in this class contain information about the vertices, indices, normals, texture coordinates, as well as a information of what material is related to that geometry.

LibraryMaterial class contains a reference to the effects which are defined in the LibraryEffects class.

LibraryEffects is a class that contains the effects. Effect contains references to color or texture. Otherwise, it is possible that the material has set the texture and colors together.

COLLADA parser design required a special optimization and reason is the specificity or the format and capabilities supported by graphics hardware. So, COLLADA library_geometries element is very specific in terms of indices connectivity and the one-dimensional field defined in the source elements. Indices defined in this format are defined as one-dimensional field. This field contains the indices of point coordinates, normals, texture coordinates. If the models are presented with basic primitives, (triangles), then there is an element named triangles into the library_geometries element that contains the index fields as the input elements with semantics, source and offset attributes that determine which indexes are indices of points, normals and texture coordinates. Otherwise, the graphics hardware for visualization uses OpenGL ES 2.0, which supports a unique index field common to all these types of coordinates. The parser successfully converts the geometries of the COLLADA format in the geometry of the joint field indices, which are placed in mesh class as shown in Figure 1. This conversion is presented in Figure 3.
Results

This section describes the evaluation of the parser (importer) for the COLLADA format, where the conversion of 3D content into the mesh was successfully completed. A base class for XML parsing was created, and its methods and attributes are inherited from the collada.Manager class. The implemented class adapted with the COLLADA format and adopted with its methods successfully obtain the necessary information (geometry, materials, normal) for 3D visualization to the Web browser. During the development of this library, it was performed the optimization and data adapting from this format with the graphics hardware.

This format has all the elements for the description of complex 3D scenes, and the visualization of the Web browser is enabled using WebGL technology. WebGL is a direct mode for 3D rendering, or API designed to represent 3D data in the Web browser. It is based on OpenGL ES 2.0 and provides similar functionality for the rendering, but in the context of HTML (the HTML Canvas element). In this way the use of plug-ins for rendering 3D scenes on the Web browser and software rendering is avoided, but it leave rendering to the graphics hardware. The limitations of this technology are: OpenGL | ES2.0 and WebGL do not support 32-bit values for the vertex indices. WebGL supports the identifier name of no more than 256 characters. The maximum number of vertices per chunk is 64K. Figure 4 presents the results of the use COLLADA parser (importer's). Performance and characteristics of the model described in the COLLADA format are determined by several parameters. For each model it is calculated: the download time, the total model parsing time, the number of the mesh model, the number of triangles, number of vertices, number of materials, number of textures, number of indices, the number of frames per second.

These data were obtained by testing the library described in the following Web browsers Firefox 4.0 beta 11, Google Chrome 10, 11 Opera, where are calculated their average value. The tests were conducted on the platforms Windows 7 64 bit and 64 bit Ubuntu 10:10 (Linux). The values of these parameters are shown in Fig.5, while the FPS for an empty scene is 1.25fps. Configuration of the testing computer was: Intel GeForce8400M G 256 MB, Intel Core 2 Duo 1.5GHz.

<table>
<thead>
<tr>
<th>Model/Parameters</th>
<th>download time (s)</th>
<th>parsing time (ms)</th>
<th>number of triangles</th>
<th>number of vertices</th>
<th>number of materials</th>
<th>number of textures</th>
<th>number of indices</th>
<th>frames per sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health Center</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambassador</td>
<td>3</td>
<td>81</td>
<td>4</td>
<td>80</td>
<td>760</td>
<td>1</td>
<td>18</td>
<td>240</td>
</tr>
<tr>
<td>Court</td>
<td>2</td>
<td>376</td>
<td>382</td>
<td>2586</td>
<td>7995</td>
<td>115</td>
<td>148</td>
<td>8420</td>
</tr>
<tr>
<td>Nikolajev</td>
<td>11</td>
<td>2381</td>
<td>3287</td>
<td>19720</td>
<td>76640</td>
<td>6</td>
<td>1040</td>
<td>59160</td>
</tr>
<tr>
<td>Federation</td>
<td>32</td>
<td>2060</td>
<td>16</td>
<td>1357</td>
<td>75482</td>
<td>230</td>
<td>40</td>
<td>4071</td>
</tr>
<tr>
<td>Post</td>
<td>18</td>
<td>2185</td>
<td>9</td>
<td>1096</td>
<td>114476</td>
<td>1</td>
<td>222</td>
<td>3288</td>
</tr>
<tr>
<td>Mileseva</td>
<td>20</td>
<td>20528</td>
<td>16</td>
<td>4804</td>
<td>1034436</td>
<td>7</td>
<td>808</td>
<td>14412</td>
</tr>
<tr>
<td>Colosseum</td>
<td>4</td>
<td>1637</td>
<td>376</td>
<td>9988</td>
<td>174176</td>
<td>1</td>
<td>147</td>
<td>29964</td>
</tr>
</tbody>
</table>

Fig.5 Performance table
5 Conclusion
COLLADA is an industry standard. Implementation of this format required about thousands of hours of experts and experience in various fields such as CAD (Computer-Aided Design), visual simulation, animation, virtual-reality and so on.

This format is focused on quality and full support DCC tools, as well as the increasing number of tools which support this standard. Now X3D standard has almost identical specification as COLLADA for Physics. The goal of the Khronos Group is to ensure that this format is not be only interoperable but also to significantly increase the impact of the transfer of 3D models for interactive applications around the world.

Evaluation of the parser for the COLLADA standard has enabled the creation of new approaches and strategies for visualization of 3D data with the Web browsers. The advantages of this parser derived from it’s own architecture and from the COLLADA format. Applications that use this parser would allow a set of capabilities such as transport complex data via the Internet, the completeness of information that is needed for the presentation of complex 3D scenes, structural arrangement of data, data transfer and description of 3D models in a single file, data sharing with different information systems as well as providing of the transfer of interactive models and simplifed access to support and 3D data management.

Further development of this library will be directed to cover a large number of functionality and effects of the graphic engines, it includes the physics models that described in the famous portable format such as COLLADA format. The format enhancement with coming of new versions and the addition of new formats such as X3D[10], VRML, KML, DWG, DXF, U3D will affect on the compatibility of the library with other known systems. In addition there will be a conversion from one format to another and all formats will be converted in one universal format. The transfer of the compressed data is very important in communications and information systems and this could be one of the future features of this library. Further work will be focused on the implementation and expansion of the parser that will provide the methods such as: creating composite images that are composed of multiple textures for a model or part of the model, then geometry composition (vertex composition, normal composition, color composition, composition of texture coordinates), better organization of the indices for the geometry and for the texture. Further optimization of the parser should provide a much better system performance , lower memory consumption and processing time, as well as a greater use of powerful graphics hardware.

References
[9] Shaders: http://www.opengl.org/documentation/gls1