Research on Web3D in Distributed Monitoring and Control Systems

ZHANG Xinglin, DU Yuyue
Shandong University of Science and Technology
Qingdao, 266590, China
E-mail: yydu001@163.com

JIANG Xia
Laiwu Vocational and Technical College
Laiwu, 271100, China

***Abstract*—Web3D technology, with the advantages of realistic scene simulation and intelligent interaction, has been widely applied to many fields, like electronic business and medical treatment. Aiming at the problems of abstract information and poor interactivity in traditional Distributed Monitoring and Control Systems (DMCS), a method based on Web3D technology is presented. The services published by DMCS are called to deal with the request before the scene management, and then the effective messages are asynchronous transferred by the way of Ajax which can optimize the real-time scene interaction. The objects designed by JavaScript map the messages into entities in VRML/X3D scenes. Simulation analysis demonstrated the effectiveness of realistic scene simulation and high performance of intelligent interaction.**

Keywords-Web3D technology; VRML/X3D; services; DMCS; Ajax

I.INTRODUCTION

With the development and integration of computer network technology and simulation technology, the distributed simulation based on network has become an essential research field. Visual monitoring technology with the features of good effectiveness is applied in the industry area, and Distributed Virtual Monitoring and Control System (DVMCS) based on network is one of the most comprehensive research [1]. Traditional monitoring systems widely utilize the ways of 2D figures and videos that have a great deal of drawbacks. The two dimension simulation without visualization and reality cannot describe the layout and status of the realistic scenes, while the videos cannot express the complicated relations between events and the scope, angle and video clarity are easy to be effected by the environment. Besides, the data that need to be transmitted on the limited network are too big. VR (Virtual Reality) is a good choice and an effective technology to construct a platform with high performance [2].

VR, a new technology which takes computer technique as the core, is combined with computer graphics technique, sensor technique, human-computer interaction technique, network technique and others. The major feature is to construct a digital environment with the hardware, software and sensors. The publication of Web3D standard made it possible to implement VR on the Internet. The Web3D technology that includes related languages, protocols and software development tools, is a new developing method to construct 3D virtual reality [3-5]. Web3D is similar to other web techniques using the same concepts -- the content is stored in the web server, the clients send requests to get the response data that tend to be displayed in browsers. In addition, this solution can integrate those existing web applications tightly. Considering characteristics of Web3D technique, this paper chooses VRML (Virtual Reality Modeling Language) as the major research technique, thus integrating VRML with web services to achieve the network-based DVMCS.

I. SYSTEM ARCHITECTURE

DVMCS based on Web3D merges with web services published by existing systems, the duty of which is to deal with different requests from clients and to maintain the consistency of virtual world. On the other hand, human-computer interface and VRML/X3D interface serve the functions of simulation of dynamic scene and interaction with clients. Fig. 1 shows the system architecture diagram.

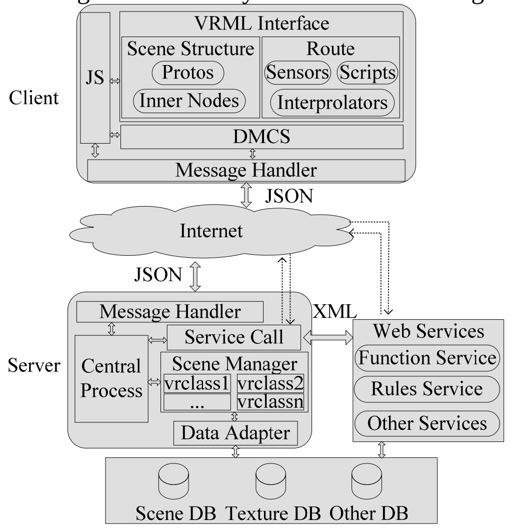

Figure 1. Architecture of DVMCS

Web server is in charge of the running of the whole distributed system, including Central Process, Service Call, Scene Manager and Data Adapter. In receiving requests from clients, the Central Process module analyses the data and calls corresponding services to process, and then the messages responded are sent to the clients. To heighten the efficiency, extensibility and platform independence of system, many functions are designed as web services for the Central Process to retrieve and assemble when needed. The duty of Scene manager module is to initiate the virtual scene and to maintain the entities consistent with the data in database. Data Adapter module provides valid data for other business modules to cope with.

The clients' browsers display virtual world, interact with users directly as well as update entities in the world according to messages received from the Web server. Therefore, the most important part of system is the VRML interface, which has essential effect on the scene information and interaction experience with users. Actually, VRML interface has to add, delete or modify the status of entities, listen to the events outputted from the whole scene and respond to them through the way of routes and functions. In virtual scenes, entities contain visible ones like buildings, and invisible ones like sensors. All of the entities are instances of vrclasses dynamically defined by the system. They have their unique identities, attributes and functions that are designed to meet the need of input or output events. These methods guarantee that initialization and update of virtual scenes, dynamic consistency of entities and the web server data are well performed.

During the process of real-time data access and scene simulation, the concept of object-oriented technique is put into practice. OOP (Object-Oriented Programming) is a method to package data and codes. Class, encapsulation and inheritance techniques are used in the structure that can make full use of the benefits of object orientation, such as hierarchy and polymorphism. In traditional systems, the servers do the work of business logic, and the clients show messages to users. Not only does DVMCS based on Web3D process the data in the server, but also converts results into highly efficient serialized messages before transmitted to the client. After that, the client is going to transform messages into attributes of objects defined by JavaScript, which can help to operate entities in the virtual scenes and interact with users. The concept of object orientation is realized in both the server and the client, particularly in the 3D interaction.

II. RELATED WORK ON SYSTEM IMPLEMENT

Pietro Murano did a great deal of research to demonstrate that the models of VRML/X3D scenes are superior to other schematic diagram in terms of location, especially when users do not know the layout of the environment [6]. VRML/X3D scenes contribute to arriving at expected locations and positioning targets, the practicality of which plays an essential role in the scientific research, simulation training and fire drill. In DVMCS, they can help to observe the whole working site from different angles, listen to events happened in the spot, and make emergency action plans to avoid the accident happen.

A. Scene Generation

The feel of realistic experience depends on those visible entities to render the virtual scene. All of the factors -- the level of model details ,the clarity of texture and images, audios and videos in the background -- can be improved to achieve a better scene at the cost of increasing network latency [7,8]. In view of the real-time performance of the system, the modeling process has some rules to comply with as follows: firstly, basic shapes are preferable, complex models are got by Boolean operation, and remove redundant vertexes and edges without reducing the model details to cut down the number of facets and the volume of file; secondly, compared with meticulous models, models that adopt texture images can not only render vivid scenes, but also reduce the complexity of modeling. The JPEG images have good features of avoiding distortion and outstanding graphic effect; finally, unique name models effectively, optimize the scene hierarchy and divide models into groups. These rules help to edit the scripts and functions to respond to events timely.

On the other hand, the system focuses on the real-time monitoring, which includes implement of dynamic entities and interaction [9]. In order to manipulate entities in an object-oriented fashion, one method that is similar to the concept of class of OOP is designed to encapsulate attributes, input/output events, routes and predefined functions into different entities. The method can save bandwidth resources, define prototype structures and instantiate entities when needed. Every instance is a duplication of predefined prototype structures and has its unique attribute values and underlying events. To manipulate entities dynamically, a PROTO nodes library is needed, and the example of the definition for NewNode nodes is as follows.

```
#VRML V2.0 utf8
PROTO NewNode [
    field SFString id ""
    field MFNode node []
    field SFBool enabled TRUE
    field SFFloat transparency 0
    field SFRotation rotation 0 0 0 1
    field SFVec3f boxCenter 1 1 1
    field SFVec3f boxSize 1 1 1
    eventIn MFNode set_node
    eventIn SFBool set_enabled
    eventIn SFBool nodeIsSelected
    eventIn SFFloat set_Transparency
    eventIn SFRotation set_Rotation
    eventIn SFBool nodeIsSelected
    eventOut SFBool enabled_changed
    eventOut SFString selectedId
    eventOut SFVec3f translation_changed
    eventOut MFNode node_changed ]
{ Script {
    url "NewNode.class"
    field SFString id IS cid
    field MFNode node IS node
    field SFBool enabled IS enabled
    eventIn MFNode set_node IS set_node
    eventIn SFBool set_enabled IS set_enabled
    eventOut SFBool enabled_changed IS enabled_changed
    eventIn SFBool nodeIsSelected IS nodeIsSelected
    eventOut SFString selectedId IS selectedId        }…}
```

The *NewNode* node has a field *id* as its own identity in the virtual world. The interactive operations and input events will lead to the state change of node, thus affecting the entities in scenes. The enabled field changes availability of entities, the transparency field changes visibility and the rotation field adjusts entities to get a right position in scenes, etc. When the *NewNode* node is selected, the *nodeSelected* field gets an input event. The *selectedId* field generates an output event when the *NewNode* node is selected, and the node will be manipulated according to the *id*. The input events, like *set_Transparency* and *set_Rotation*, are set to change the entities, and the output events, like *translation_chaned* and *node_changed*, are set to make appropriate reactions to the outside. To get a more realistic scene simulation, the work of designing entities' fields, events and functions at length is necessary.

The parameters got from outside programs must be mapped into corresponding nodes' fields, so the first step is to get the right node in line with the *id*. This paper proposes the *Id4Node* node to finish the work, and its code is as follows.

```
PROTO Id4Node [
     eventIn MFString ids
     eventOut MFNode nodes ]
{Script {  url "Id4Node.class"
           eventIn MFString ids
           eventOut MFNode nodes
           }   }
```

As the data from the database or sensors is upgraded in the web server, or the scene in the client generates events, the server transforms the result into serialized message in the format defined in the messages protocol. In receiving messages, the JavaScript functions and VRML/X3D interface functions work together to update the virtual scenes in time, for example, *setNodeEventIn()*, *getNodeEventOut()*, *create_VrmlFromString()* and functions defined in Script node.

B. Dynamic Interactive Data

For the sake of the heterogeneity of systems, distributed systems have to be compatible with different software development tools and environment. Apart from that, expansibility of systems needs to be taken into account to avoid causing bottlenecks to system scale and module extension. Web services based on XML technology can encapsulate information and functions uniformly and exchange messages among different systems regardless their own platform environment and programming languages. In DVMCS the retrieve and compose of web services is to make the most of computing resources and computing power of existing applications.

Ajax (Asynchronous JavaScript and XML) is a technique that can communicate asynchronously with servers. The method obtains required data to reduce request redundancy efficiently, lowers the burden on each server and the burden of messages transmission on the Internet, as well as fully utilizes the idle computing resources in the client [10]. Ajax advocates XML as the exchange data format, but XML organize data by redundant structures, which is weak on storing efficient data. Additionally, the grammar of XML is blurring. It is also much too time-consuming to be parsed. Thus, in some way it is not appropriate to be applied to communication with the client.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. Not only is it convenient for humans to read and write, but also is easy for machines to parse and generate. It is based on a subset of JavaScript, so it is very efficient to parse messages based on JSON in the client. The examples of object containers in XML and JSON formats are as follows.

```
<container>                        {“containers”:[
 <id>BAFU8903991</id>              {
  <enable>true</enable>             “id”: “BAFU8903991”,
  <transparency>0                   “enabled”: “true”,
  </transparency>                   “transparency”: “0”,
  <rotation>01 1 1</rotation>       “rotation”: “0 1 1 1”,
  <xh>BAFU8903991</xh>              “xH”: “BAFU8903991”,
  <cc>20</cc>                       “cc”: “20”,
  <xx>TK</xx>                       “xx”: “TK”,
  <tdh>QDST011528</tdh>             “tdh”: “QDST011528”
</container>                       }]}
```

In monitoring and control systems, enhancing efficiency of messages and lessening the burden of transmission and parsing time can improve the systems dramatically. The tests prove that JSON is a better choice as data-interchange format. Table 1 lists the performance comparison of the two data format (2000 container objects taken as samples).

TABLE I. COMPARISON OF XML AND JSON

Format	**Size (KB)**	**Download Time(ms)**	**Parse Time(ms)**	**Total Time(ms)**
XML	484	493.1	92.1	585.2
JSON	332	345.4	23.6	369.0

Owing to that JavaScript has not the ability of defining Class, the objects data parsed from messages can only be stored into a non-reusable object, which just needs to be created and to initialize variables. Without using the concept of Class, the object instanced cannot be verified whether it is a container object or not. To achieve the requirement of standard object definition, a new method is introduced. A basic function is designed as basic class, and it is not going to run when called but to invoke its sub-functions. The sub-functions define attributes and programs that can access variables of basic function. For example, the code of container class designed in JavaScript is as follows.

```
function Container(cId,cEnabled,cTransparency,cRoation)
{ var id=cId;     var enabled=cEnabled;
   var transparency=cTransparency;
   var rotation=cRotation;
   this.set_id=function(cId){id=cId;}
   this.get_id=function(){return id;}
   this.set_enabled=function(cEnabled){enabled=cEnabled;}
   this.get_enabled=function(){return enabled;}
   this.get_transparency () {return transparency ;} ……}
```

The variables, such as id and enabled, can only be accessed inside Container function. The inside functions (set_* & get_*) can be called to access those inside variables by outside program. Object oriented programming is realized in the script program by this way, which contributes to interacting with entities of virtual scenes.

III. SYSTEM REALIZATION

DVMCS was adopted by Qingdao Port to monitor dangerous goods container yard, and it enhanced the working efficiency of existing system effectively. The yard constructions and facilities were modeled in accordance with CAD drawings before achieving the whole layout of virtual scene. To render the scene as realistic as possible, texture and material images were collected on the spot. Then, the organization structure of VRML file was improved, which made it convenient to add interaction functions to nodes. The system simulated a realistic yard scene and dynamic states of dangerous goods containers, and the data obtained from container services and sensors were classified, transmitted and mapped into entities in scene timely.

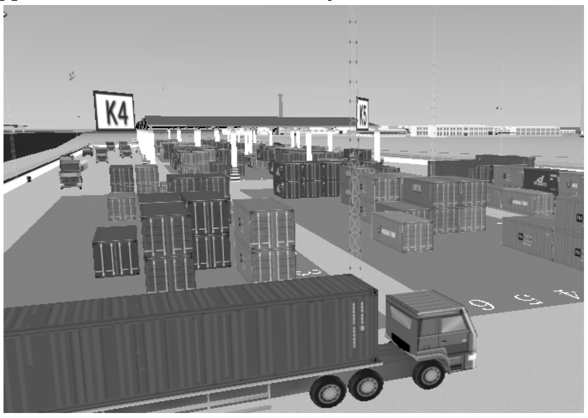

Figure 2. Yard monitoring in DVMCS

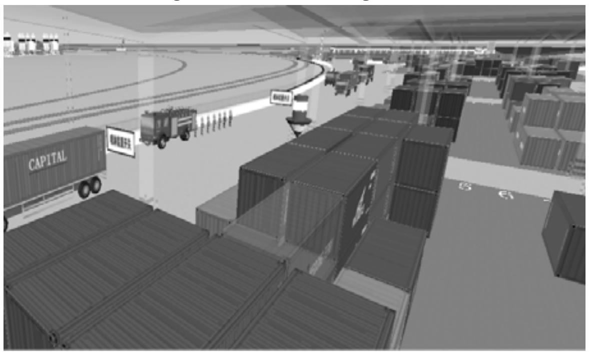

Figure 3. Dynamic entity locating in DVMCS

Fig 2 and Fig 3 show the running effect of DVMCS. To meet different requirements of monitoring, users have several optional monitoring patterns to choose. Under the right pattern, users are able to supervise every yard, just like on the spot, and to pass through every object to observe targets in different distance and angles. Using the information and layout of containers can check whether the position of containers obey the rules of dangerous goods management. The target container can be located and presented remarkably. There are many practical functions designed and tested in the system, and the system application shows that DVMCS is easily accepted and supported by users owing to its friendly 3D virtual environment and outstanding interaction design.

IV. CONCLUSION

This article proposes a novel approach, integrated with published information services, to implement distributed monitoring and control system based on Web3D. We came up with steps to optimize modeling process and raised operability and consistency of entities. Meanwhile, we incorporated object oriented technique into the interaction with dynamic virtual scene. The problems of data access and collision detection are also improved. As an emerging technology, applications based on Web3D require further research for better usability and accessibility.

ACKNOWLEDGMENT

This project was supported by National Natural Science Foundation of China (61170078) and Technological Developing Program of Shandong Province of China (2011GGX10114).

REFERENCES

[1] Zuoyi Duan, Wei Wu. Qinping Zhao, “Component-Based Distributed Virtual Reality Application System,” Journal of Software, vol. 17, no. 3, pp. 546-558, 2006.

[2] Zhao Li,” Virtual Marine Environment Spatio-Temporal Data Modeling and Service for Visualization,” Hang Zhou: Zhejiang University, 2010.

[3] Minrui Fei, Xiaoli Wang, Lixiong Li, Tao Wang, “ Som New Development in Remote Monitoring and Control Systems Based on Virtual Reality,” Journal of System Simulation,” vol. 20, pp.386-390,2005.

[4] Wei Lei, “Collaboration in 3D shared spaces using X3D and VRML ,” 2009 international Conference on CyberWorlds, 2009, pp. 36-42.

[5] Lucio Ieronutti, Luca Chittaro, “ Employing virtual humans for education and training in X3D/VRML worlds,” Computer & Education, vol. 49, no. 1, pp. 93-109, 2007.

[6] Pietro Murana and Dino Mackey, “Usefulness of VRML building models in a direction finding context,” Interacting with Computers, vol. 19, no. 3, pp. 305-313, 2007.

[7] Zhang Xiushan and Wu Chanle, “Hierarchy-invariant interactive virtual maintenance model and its VRML-based simulation,” 2009 WRI World Congress on Computer Science and Information Engineering, CSIE 2009, vol. 2, pp. 115-120.

[8] Jinyuan Jia, Guanghua Lu and Yuan Pan, “An efficient navigation algorithm of large scale distributed VRML/X3 D environments,” , Lecture Notes in Computer Science, LNCS, vol. 3434, pp. 244-252, Springer-Verlag Berlin Heidelberg, August 2007.

[9] Haiqing Li, Guofu Yin and Biyou Peng, “X3D-based Interactive Transformer Substation Information Visualization Management System,” Computer Engineering, vol. 33, no. 3, pp. 265-267, February 2007.

[10] Dave Crane and Eric Pascarello, Ajax in action, Manning Publications, 2006.