Global Journal of Computer Science and Technology
Graphics & Vision
Volume 12 Issue 12 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

3D Interfaces for Real Time Monitoring of Radwaste Storage

By Gianfranco Vecchio & Paolo Finocchiaro
Istituto Nazionale di Fisica Nucleare

Abstract - This paper describes the design and development of a 3D interface in the framework of an application for the monitoring of a radioactive waste storage. It focuses on the description of software solutions by integrating different technologies. We used only free and open-source libraries to develop the 3D environment that were subsequently integrated into the existing web application developed used Java Server Faces. In order to implement 3D graphics we used Away3D, an open source 3D graphics engine for Adobe Flash, written in ActionScript 3 and runnable in every browser that utilizes Adobe Flash Player.

Keywords : *Radiation monitoring, Radioactive waste, Interactive systems, Graphics Animation, Graphical models.*

GJCST-F Classification: *I.2.10*

Strictly as per the compliance and regulations of:

3D Interfaces for Real Time Monitoring of Radwaste Storage

Gianfranco Vecchio [α] & Paolo Finocchiaro [σ]

Abstract - This paper describes the design and development of a 3D interface in the framework of an application for the monitoring of a radioactive waste storage. It focuses on the description of software solutions by integrating different technologies. We used only free and open-source libraries to develop the 3D environment that were subsequently integrated into the existing web application developed used Java Server Faces. In order to implement 3D graphics we used Away3D, an open source 3D graphics engine for Adobe Flash, written in ActionScript 3 and runnable in every browser that utilizes Adobe Flash Player.

IndexTerms : Radiation monitoring, Radioactive waste, Interactive systems, Graphics Animation, Graphical models.

I. Introduction

This document wants to show a new useful way to display information from data collected in a harsh environment. An electronic system for radioactive waste monitoring was developed at INFN-LNS laboratory, as part of DMNR project [1]. It consists of a network of sensors deployed around radwaste drums for the continuous detection of radiation [2]. The sensors employed so far, specially developed for this case, are made of plastic scintillating fibers whose light is readout by means of pairs of Silicon Photomultipliers (SiPM). A counting system based on FPGA boards handles the data flow coming from groups of these sensors and sends it to the front-end of the application. A relational database system is used to store the historical data, and a dynamical web application was developed to explore the database and display the corresponding data in graphical form.

To improve the view of the storage following a database query we developed a 3D scene that shows the status of the whole plant at a glance [3], [4]. We developed two kinds of 3D scenes: a representation of the whole plant with the possibility to navigate through the drums, and the detailed view of a single structure of three drums with fiber sensors around them. In this document we are going to describe a setup with up to three drums inside tower-like structures and arranged in chessboard fashion.

The first scenario is used to investigate the status of the repository in a user-selected time range, and to inspect every single drum by virtual navigation through the plant using mouse and keyboard, as shown in Fig. 1.

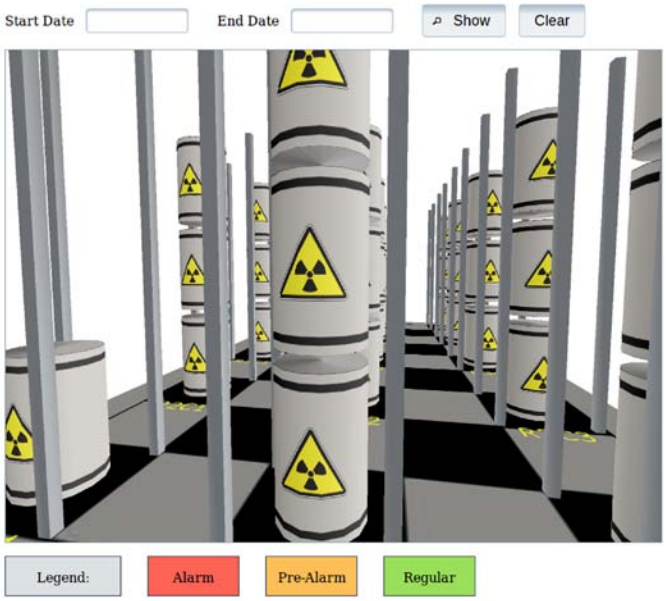

Fig. 1 : 3D view of radioactive waste repository

Once the user fills the date textboxes data coming from the database will color the drums in red, orange or green according to the corresponding alarm condition (Fig. 2).

Should there be no data in the database for the chosen time range, the drum will be drawn as grey. The colored drums can be clicked on by the user, in order to have a detailed view of its status (i.e. of all the sensors surrounding it).

The second scenario represents the detail of a single box containing the drum clicked in the previous scenario. It also draws the fiber sensors around each drum (four horizontal fibers in this sample case), and the interactivity consists on the possibility to rotate the tower to gain visual access to all the fibers and their alarm status. Clicking on a fiber one can get the plot of its counting rate in the time range selected. If one clicks on a drum the collective plot for the four fibers of that drum is shown in the same chart (Fig. 3).

Moreover, clicking on a drum a polar chart of the counting rate for each sensor will be shown, along with a chart indicating maximum, minimum and average counting rate values.

Author α σ : INFN Laboratori Nazionali del Sud, via S. Sofia 62, 95125 Catania, Italy.
E-mail α : vecchio@lns.infn.it
E-mail σ : finocchiaro@lns.infn.it

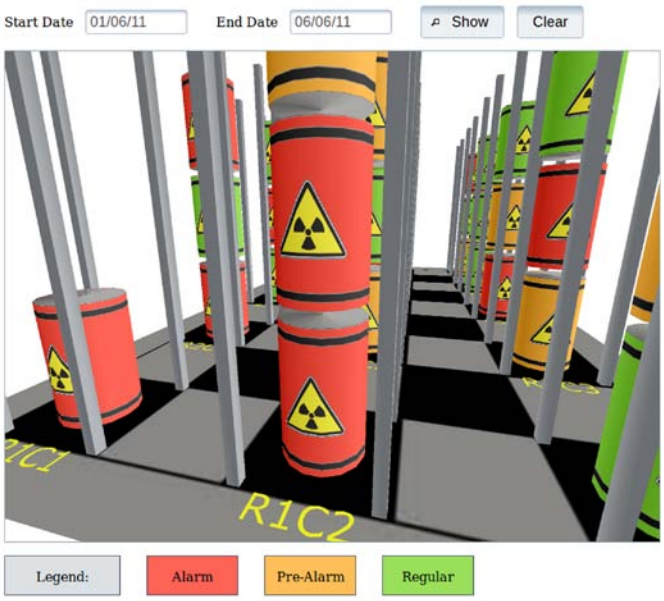

Fig. 2 : 3D view of radioactive waste plant after database query

II. Away3D, A Library for Adobe Flash to Create the Storage Plant

In order to simulate the environment that we have to monitor as well as possible, we chose Adobe Flash for its powerfulness in the creation of complex interactive animations. Away3D [5]-[8] is a 3D graphic engine written for Adobe Flash platform in ActionScript 3 [9] originally derived from Papervision3D. It was designed to be fast and extensible and allows the easy creation of various 3D objects with the use of its graphics primitives. Working on 3D coordinates and leveraging the object oriented nature of the language, we were able to build a realistic scene of our plant. We developed our project with Flash Develop [10], an open-source Integrated Development Environment to write and to compile Action Script code and to create Shockwave Flash objects (SWF).

The navigation inside the nuclear repository is done by means of a camera, defined by Away3D, moved using mouse and keyboard. Away3D offers several different types of camera, that represent the viewer's position in space relative to the rest of the scene. For the scene representing the whole storage we used the default *Camera3D* that produces a standard front-facing perspective projection of the scene. Thanks to event handles and objects like *MouseEvent3D* and *KeyboardEvent*, we were able to add some interactivity and better user experience. Hence, like in a computer game, by using the mouse and the arrow keys (or W-A-S-D keys) one can walk and look around every drum inside the environment.

As for the scene regarding the single tower box, which can be rotated using the mouse, we employed a different camera, the *HoverCamera3D,* that allows the user to navigate each side of the box only changing some properties like distance and rotation angles.

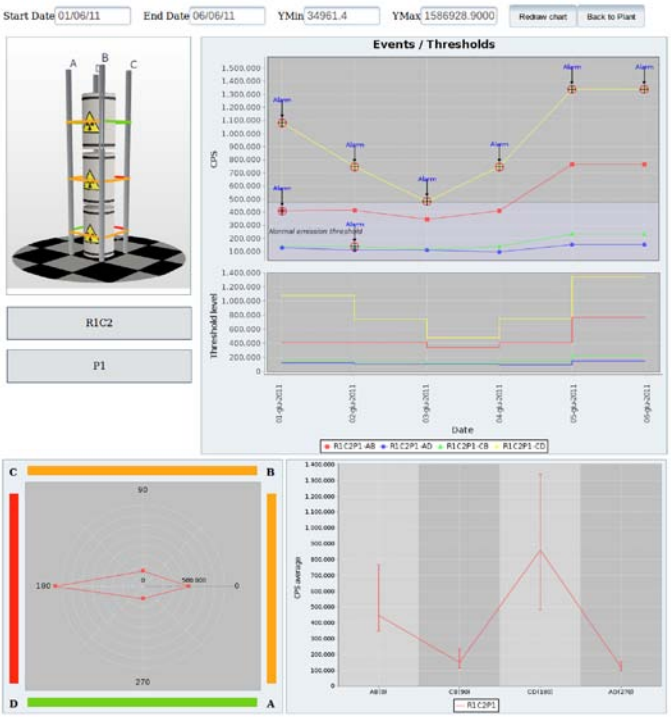

Fig. 3 : 3D view of a box with detail of the four sensors. Clicking on a fiber or on a drum one can view the global counting rate chart

Each object rendered to the scene was built from geometric elements like triangles and line segments. To make this mesh more realistic we had to build images to be set in the material property of each element. To make towers, drums and fibers we used the same Away3D primitive, the *Cylinder*, and then we used different material for every type of element.

Fig. 4 shows the result of the storage plant before the application of materials in each element. We can see only the mesh that was used to build every object made by triangles and line segments, showed by default in different color by the 3D graphic engine.

Furthermore, we were able to write text inside the interactive scenes, embedding the font inside an SWF object. We used text for orientation purposes and to view the ID of each element.

Finally, the possibility to pass some parameter to external resources, was achieved thanks to flash class *ExternalInterface* that enables straightforward communication between ActionScript and SWF containers, that in our case consisted of the XHTML page with JavaScript:

```
ExternalInterface.call("javascript_function_to_cal
l", parameters_to_pass_to_javascript);
```

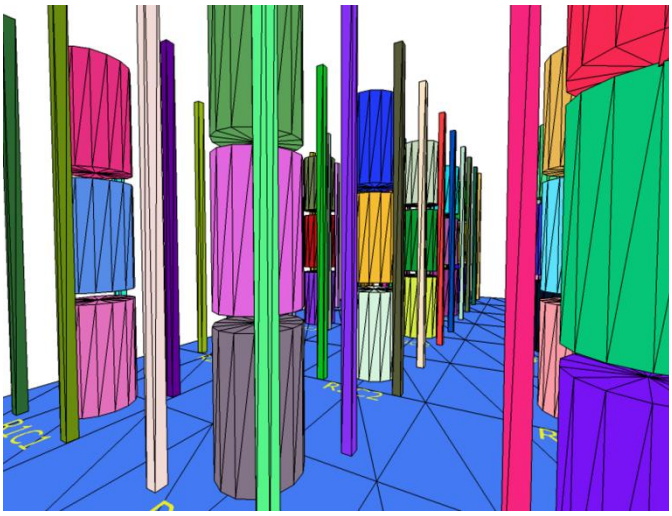

Fig. 4 : Mesh of elements inside the storage plant

III. Integration Between Different Technologies: Adobe Flash and Java Server Faces

The most complex task was represented by the encapsulation of the flash object inside the existing Java Server Faces [11], [12] project and the data exchange between the two different technologies [13]. The elements in the storage plant have to change according to the data coming from the database. Our web application contains a Java classes (the Managed Beans) that encapsulate the state of an object, and then stores all the data coming from the database after a user query. To pass data between Managed Beans and SWF objects we used JavaScript in conjunction with *composite components*, introduced after 2.0 version of JSF. These allow to create reusable components and to embed external objects easily.

In our application we defined two composite components, one for each SWF object. They are simply XHTML files, implemented in JSF fashion, containing an interface and an implementation. JSF uses namespaces to include new tags to be used in the XHTML page, and for composite components the namespace is:

```
xmlns:cc="http://java.sun.com/jsf/composite"
```

The interface contains configurable features, like input and output data, whereas the implementation contains JSF tags needed to build the new component.

Inside the interface we declared variables that are needed to pass parameters between applications through the *attribute* tag:

```
<cc:attribute name="drums">
```

In the implementation, instead, we embedded the flash object thanks to JavaScript and the open source script *SWFObject* [14]. It offers methods to insert flash contents in HTML and XHTML pages, using only a small JavaScript file. It is more optimized and flexible than any other currently available method for the insertion of flash content and it is easy to use. Indeed, we need to use only this function to embed our SWF object:

```
swfobject.embedSWF(swfUrl, "800", "600", "11.1.0", flashvars);
```

This method accepts as parameters the URL of an SWF file, the size of the window that will show the flash content, the flash player version, and the *flashVars* variables that include data from Managed Bean.

Fig. 5 shows the schematic approach of this integration, describing the use and the configuration of composite components.

In order to use a composite component we only need to declare a namespace with the name of the directory that contains it:

```
xmlns:cc="http://java.sun.com/jsf/composite/flash"
```

and just a line is needed to embed it in our code and to pass the parameters in the classic JSF notation:

```
<cc:plant id="plant" drums="#{event3d.drums}"
    camera="#{event3d.cameraCoordinates}">
```

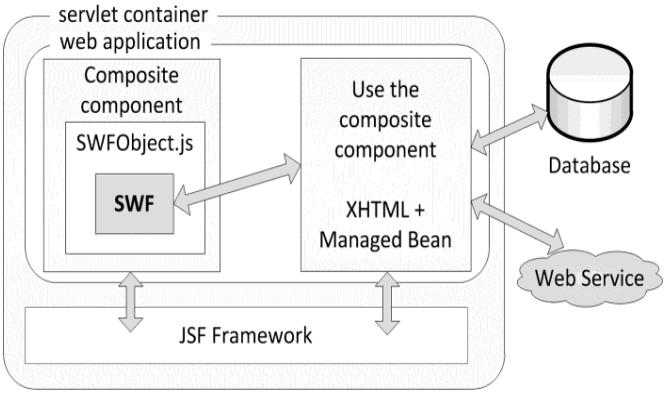

Fig. 5 : Scheme of the use and configuration of composite components

The flash object needs dynamic data to work that come from the database. As seen before, they will be passed to the flash object by means of *flashVars* variables. The attributes *drums* and *camera*, in the code shown above, are declared in the interface section of composite component and they will be used in the implementation code and inside the definition of flashVars variable:

```
var flashVars = {drums: "${cc.attrs.drums}", camera: "${cc.attrs.camera}"};
```

In order to retrieve these data, on the flash application side, we need to use only the *LoaderInfo* class from *flash.display* package:

```
var input:String = LoaderInfo(this.root.loaderInfo).parameters.drums;

var data : Array = input ? input.split(/[^a-zA-Z0-9\-]+/) : [];
```

Hence, the data structure coming from flashVars variable, will be decrypted as a string, that we changed into an array structure for convenience.

To interact with the flash object, clicking on a fiber or on a drum, we needed the implementation of a bidirectional communication system, then of parameter passing between a flash object and a managed bean. For this reason we used JavaScript like an interface between the two different technologies. As seen in the II Chapter, with ActionScript 3 we only need to use the *ExternalInterface* class to call the JavaScript function that resides in the implementation section of composite component:

```
ExternalInterface.call("clickPlant", drums, camera
Coordinates);
```

We pass two parameters to the JavaScript function: the ID of the clicked drum and the current camera coordinates to retain the position and orientation of the camera after navigation. Finally, these parameters will be associated to managed bean attributes through hidden input text called by JavaScript:

```
clickPlant = function(drum, camera){
        document.getElementById("${cc.clientId}:for
m:drumSelected").value=drum;
        document.getElementById("${cc.clientId}:for
m:cameraCoordinates").value=camera;
}
<h:form id="form" style="display:none">
        <h:inputHidden             id="drumSelected"
value="#{event3d.drumSelected}"/>
<h:inputHidden               id="cameraCoordinates"
value="#{event3d.cameraCoordinates}"/>
</h:form>
```

At this point, with managed bean attributes set, we can show the charts related to the clicked drum or fiber in the traditional manner in JSF paradigm.

IV. Conclusion and Prospects

In this paper we presented a new candidate system to show and to monitor a radioactive waste storage plant with 3D graphics interactions. It is useful to have a simulated perception of the waste storage at a glance, and a realistic way to see the elements that belong to the repository.

We investigated how to take advantage of new features in JSF 2.0, as composite components, and integrate Flash into our JSF application. These new capabilities in JSF free us from the need to take care about data encoding and decoding for transfer. Furthermore, to pass variables and parameters we used JavaScript variables and methods, that allow an easy way to exchange data between JSF and Flash.

Finally, to embed the flash content we used SWFObject, that detects the Flash Player version and determines whether Flash content or alternative content should be shown, to prevent outdated Flash plugins to break Flash content. It offers a simple way to embed Flash content and it is more optimized and flexible than any other Flash Player embed method around.

The future work will consist of the graphic representation of different sectors of larger repositories, to be shown dynamically depending on the user selection. Moreover, we will investigate about additional graphical improvements, especially with the use of the new release of *Away3D*, the version 4.0, which will completely support the release 11 of Adobe Flash Player. Indeed, the latest version of this plugin allows to take full advantage from graphic acceleration of the Graphics Processing Unit, releasing the CPU from such a heavy computational task.

A possible improvement will consist in correlating the virtual navigation procedure with a real inspection performed by means of a robotic arm, currently under development, equipped with a CCD color camera and a spectroscopic gamma ray detector [15].

A pilot DMNR system, which will include the 3D virtual navigation interface described in this paper, is going to be installed soon for tests in a real radioactive waste storage site.

V. Acknowledgment

We are indebted with L. Cosentino, A. Pappalardo and S. Scirè for their constant support and encouragement. Special thanks to S. Cavallaro, S. Pulvirenti, E. Furia and B. Diana for the fruitful discussions.

References Références Referencias

1. P. Finocchiaro, et al., "DMNR: a new concept for real-time online monitoring of short and medium term radioactive waste" in *Radioactive Waste: Sources, Types and Management*, Nova Science Publishers, in press.
2. Predisposal Management of Low and Intermediate Level Radioactive Waste, *IAEA Safety Guide*, WS-G-2.5.
3. Chengjie Gu, Shunyi Zhuang, Zailong Zhang, and Pan Wang, "Virtual Interactive Online Exhibition Architecture in E-Commerce based on Flash 3D and Flex", *2nd International Conference on e-Business and Information System Security*, May 2010.
4. Wenshan Hu, Hong Zhou, and Qijun Deng, "Design of web-based 3D control laboratory", *2nd International Conference on Intelligent Control and Information Processing (ICICIP),* July 2011.
5. Away3D. Available: http://away3d.com
6. R. Bateman, R. Olsson, "The Essential Guide to 3D in Flash", Friends of ED, June 2010.
7. M. Casperson, "Away3D 3.6 Essentials", PACKT, February 2011.
8. M. Ivanov, "Away3D 3.6 Cookbook", PACKT, June 2011.

9. Adobe, “Adobe Flex 3 Programming ActionScript 3”, 2008. Available: http://livedocs.adobe.com/flex/3/progAS_flex3.pdf
10. FlashDevelop. Available: http://www.flashdevelop.org/
11. D. Geary, C. Horstmann, “Core JavaServer Faces (3rd ed.)”, Prentice hall, June 2010.
12. Java Server Faces Technology. Available: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
13. Re Lai, “Using Adobe Flex and JavaFX with JavaServer Faces 2.0”, March 2011. Available: http://www.oracle.com/technetwork/java/lai-flex-javafx-jsf-301278.html
14. SWFObject. Available: http://code.google.com/p/swfobject/
15. Carlotta Scirè Scappuzzo, “Studio di un sistema robotizzato per il monitoraggio di depositi di scorie radioattive”, unpublished. Available: http://www. lns.infn.it/index.php?option=com_docman&task=doc_download&gid=244&Itemid=143

This page is intentionally left blank