UML of a Touch Based 3D Visualization Environment for the Semantic Information System

A. Milshteyn, S. Mendoza, G. Herman, E. Tsai, A. Lin, Adrienne S. Lam, H. Boussalis, and C. Liu

***Abstract*—This research focuses on the development and integration of the touch-based 3D visualization environment for the Semantic Information System. It allows for collaboration between multiple users in a dynamic environment, facilitating the users' ability to generate metadata and associative content in real-time. Both metadata and content are stored in a database while their semantics are represented in the XML–based tree structure display. The UML diagrams present detailed visualizations and interconnectivity of the following system components: OpenGL-based GUI, XML parser, object-generating Scraping Tool, and Qt-based widgets that comprise a traditional GUI environment. These modeling diagrams demonstrate the interactive processes between system components of the touch-based 3D and the traditional 2D GUI environments in terms of data visualization and enhanced system functionality. The visual multilevel metadata representation provides the users of Semantic Information System with operational flexibility for efficient content management and organization.**

***Index Terms*— Graphical User Interface (GUI), Semantic Information System (SIS), Unified Modeling Language (UML),Visualization Environment.**

I. Introduction

PREVOUSLY, Structures, Pointing, and Control Engineering (SPACE) University Research Center (URC) has developed a prototype of the Semantic Information System (SIS) [1]. SIS is designed from the concept of the Semantic Web, a platform where information is organized based on a machine recognizable matter [2]. In the SIS, the semantic information is generated, stored, and hierarchically organized by a collaborative group of users. Human intervention is a necessary step during the generation of semantics of information, as the information's appropriate description and categorization result in more precise and powerful search engines [3]. Moreover, the users who acquire information may also trigger the Semantic Web to categorize and reorganize such information for enhancing the search capability in the future.

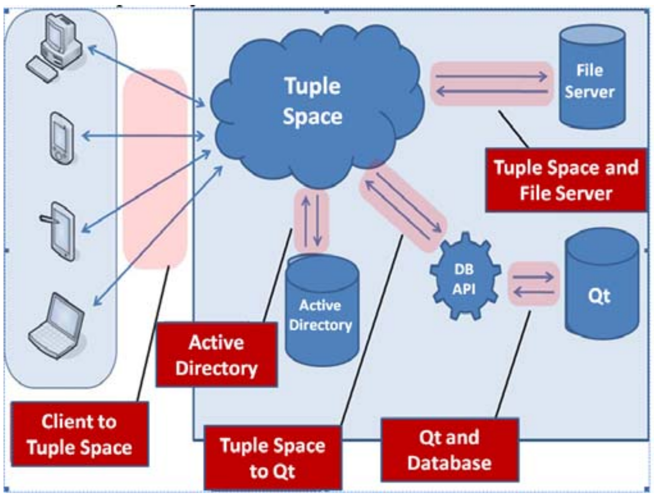


Fig. 1. Semantic Information System Architecture

This paper is focused on the UML design of an enhanced graphical visualization module with navigational capabilities for the Semantic Information System. The primary aspect of the SIS is the ability for its users to collaboratively generate new content based on the semantics of interrelated objects. The resulting project-specific database is available for further user-modifications and top-level visualization of interrelated object hierarchy. The SIS architecture is presented in Figure 1, showing the communication interactions between primary server-side components. The Tuple Space in this architectural design serves as a bridge for all the SIS intra-cluster communication [4].

The core SIS client-side functionalities are primarily composed of the object-generating Scraping Tool, an integrated web browser for content searching and a XML parser for metadata extraction [5]. These tools are responsible for generating, storing, retrieving, and displaying of semantic relationships between objects based on the XML metadata.

The UML (Unified Modeling Language) is geared for object-oriented analysis and design. It is used to model, document, and visualize individual elements of an object-oriented system. The UML presents system components in various perspectives of the project. It is commonly used in industry for designing visual models of software-intensive

Manuscript received July 13, 2013; revised August 11, 2013. This work was supported in part by the NASA University Research Center Program under Grant NNX08A44A

A. Milshteyn is with the Structures Pointing and Controls Engineering(SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (phone: 323-343-5445; e-mail: aleks.milshteyn@gmail.com).

G. Herman is with the Structures Pointing and Controls Engineering(SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (e-mail: gartheherman@gmail.com).

S. Mendoza was with the the Structures Pointing and Controls Engineering(SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (e-mail: smendoz2@calstatela.edu).

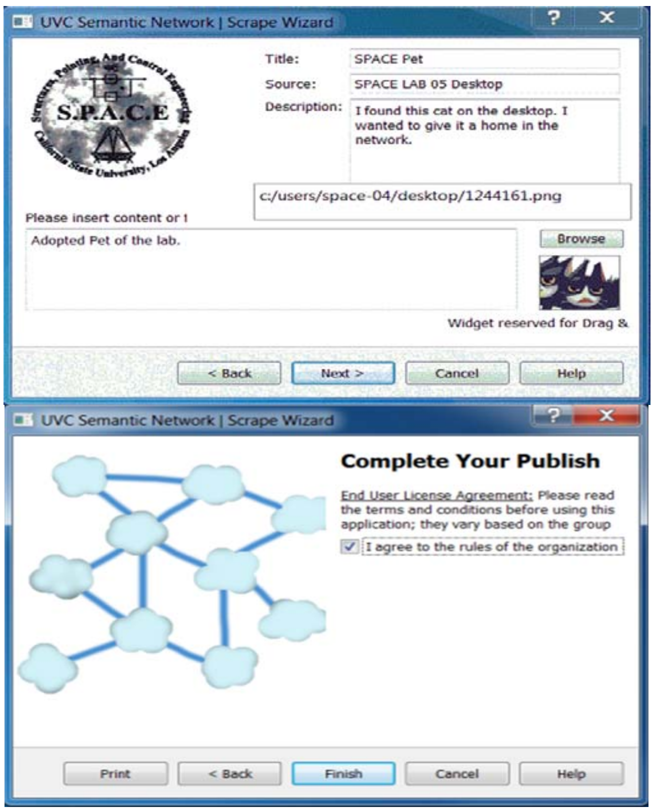
and complex systems, thus formalizing the organization of the project architecture.

The initial UML system design of the SIS Network provided assistance with visual evaluation of the system interdependencies. This top-level visualization accelerated the implementation process of new system components, as well as the enhancements of existing system components. The UML diagrams are presented throughout the paper in reference to the corresponding system modules, which are discussed as follows: Chapter 2 presents the original version of the Semantic Information System, focusing on its initial core components. Chapter 3 discusses the touch-based 3D version of the SIS, focusing on the processes of initialization, user control aspects, and texture mapping. Chapter 4 presents results of the multi-level project examples and compares the two GUI designs. Chapter 5 summarizes the need for transition from the traditional version of the Semantic Information System to the 3D touch-based navigating environment.

II. ORIGINAL SEMANTIC INFORMATION SYSTEM

A. *The Scraping Tool*

In the Semantic Information System, the mechanism for gathering content and populating the tree structure is called the Scraping Tool. Through this content gathering and publishing process, users can obtain relevant material from variety of remote sources and generate their own project based on the information collected. The Scraping Tool is initiated upon user selection and provides a pop-up menu with several steps, as seen in Figure 2.

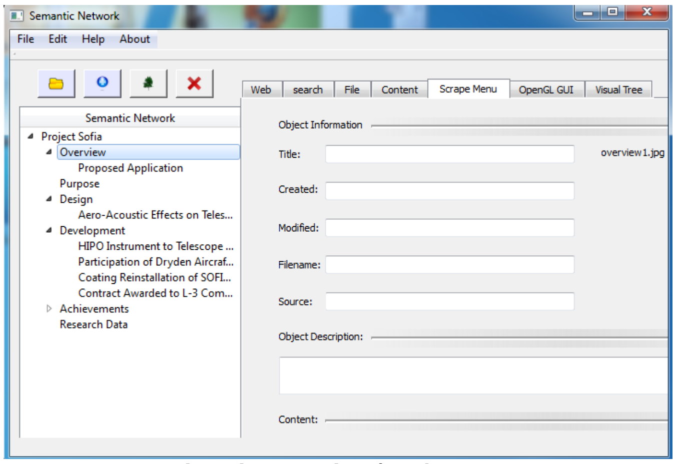

UVC Semantic Network | Scrape Wizard

Title: SPACE Pet
Source: SPACE LAB 05 Desktop
Description: I found this cat on the desktop. I wanted to give it a home in the network.
c:/users/space-04/desktop/1244161.png
Please insert content or t
Adopted Pet of the lab.
Widget reserved for Drag &

UVC Semantic Network | Scrape Wizard

Complete Your Publish
End User License Agreement: Please read the terms and conditions before using this application; they vary based on the group
☑ I agree to the rules of the organization

Fig. 2. Scraping Tool Object Generation


Semantic Network
Project Sofia
Overview
Proposed Application
Purpose
Design
Aero-Acoustic Effects on Teles...
Development
HIPO Instrument to Telescope ...
Participation of Dryden Aircraf...
Coating Reinstallation of SOFI...
Contract Awarded to L-3 Com...
Achievements
Research Data

Object Information
Title:
Created:
Modified:
Filename:
Source:
Object Description:
Content:
overview1.jpg

Fig. 3. Traditional GUI Display after Object Generation

Figure 3 shows a resulting step of the publishing process, where the object (node of information) generated by a user via the Scraping Tool application is displayed in a traditional GUI.

After a newly generated object has been published by the user, its metadata is stored in a new entry in the database, while its content (i.e. file) is uploaded on a file server. The steps involving the database and file server are transparent to the user, occurring in the background of the application and will be further discussed in this chapter. The tree structure shown in the GUI is immediately updated after the objects have been created, in order to allow users access to the modified project content.

It was previously mentioned that the accumulated content can consist of project work itself or merely the information related to the project. In either case, the SIS is more effective when the scope of its usage is well-defined by the project collaborators. This is an inherited trait of the SIS Network; as its scope becomes broader in focus, the benefits for its audience are minimized. Researching information in an unfocused SIS Network will provide insignificant improvements over a regular Internet search. The SIS network is heavily dependent on the users that proactively add content that is relevant to a project or knowledge base.

B. *Database and FTP*

The database in the SIS serves as a means for storing metadata, facilitating searching functions, and managing user accounts. The objects within the network can be searched by the metadata parameters such as date created or file type. For larger and more heavily populated networks, the ability to search within specific branches can be compounded on top of the metadata parameters.

Each object within the database has a unique ID associated with it. These IDs are present in the XML tree file and are called when a user selects the object. The metadata for that particular object is pulled from the database by using the object's ID. As the project tree is populated with new objects, their respective database IDs are added to the XML file to provide a respective reference. The complete process is represented via the UML activity diagram, which is shown in Figure 4 below.

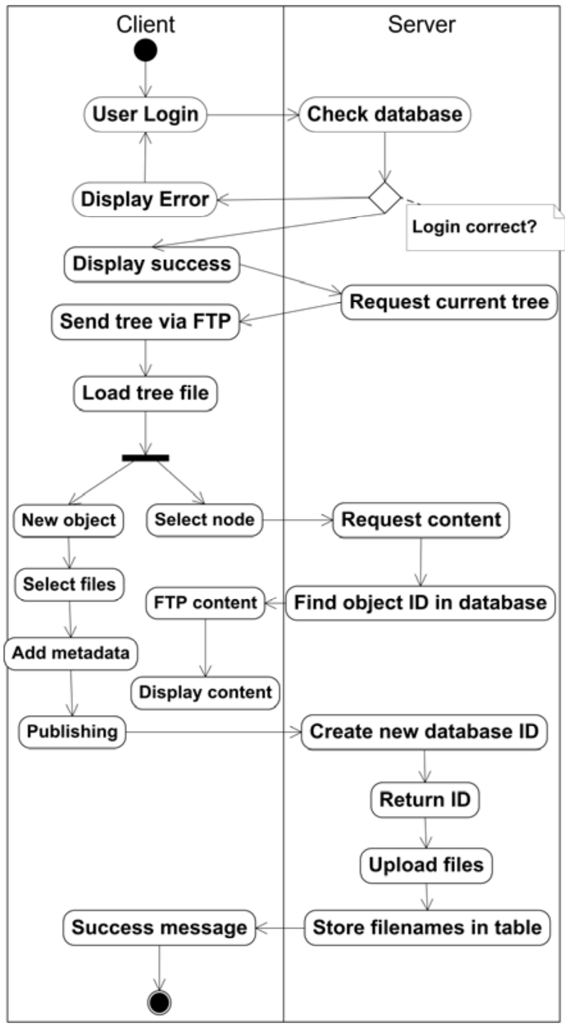


Fig. 4. UML Activity Diagram of the SIS FTP and Database Intercommunication

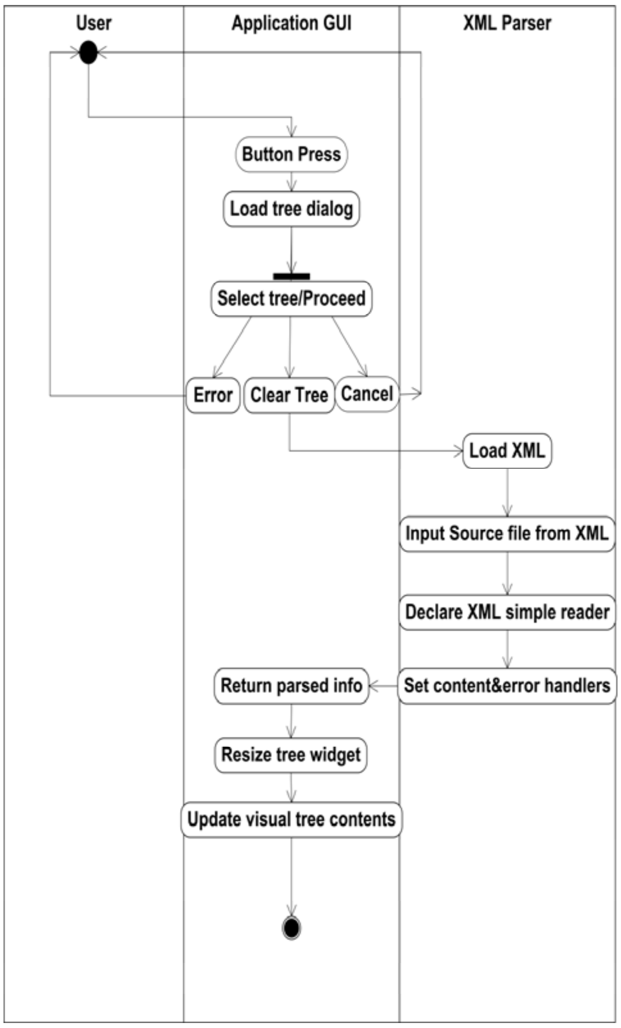


Fig. 5. UML Activity Diagram between traditional Application GUI and the XML Parser

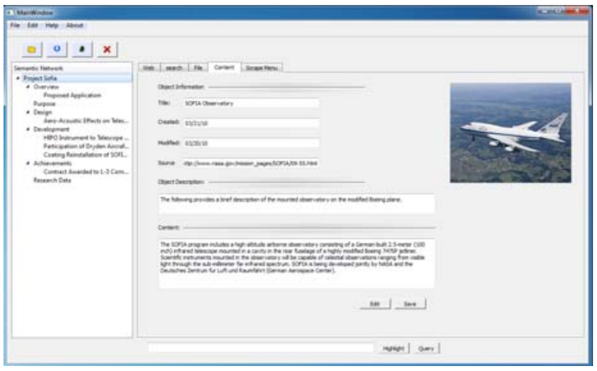
The connection between the XML-based project tree and the database is also necessary for retrieving content from the server. In addition to the aforementioned metadata parameters, the database also keeps a list of files associated with each object. This list must be embedded to the project tree in order to retrieve the object files from the FTP server. Whenever a user views or downloads an object's content, it is transparently being retrieved from a file server.

C. *The Traditional GUI Display*

This section describes the operations for the traditional GUI used in displaying a tree structure. The original GUI contains a hierarchical string-based tree structure as the sole method to navigate through the system. Such methods of navigating through files have become uncommon and have little visual impact on the user.

Figure 5 shows the activity UML diagram that depicts the user invoking the traditional GUI, which in turn communicates with the XML parser. This XML parser loads a previously user-created project tree by identifying the hierarchical level structure relationships within the project's information nodes.

Figure 6 shows a comprehensive example of a hierarchical visual tree structure that has been generated by a group of SIS participants via Scraping Tool application. On the left hand side in Figure 6, the tree structure is displayed using XML, giving the user the ability to expand and collapse the parent/child content and see what branches it contains.



Title: SOFIA Observatory

Created: 03/21/10

Modified: 03/20/10

Source: http://www.nasa.gov/mission_pages/SOFIA/SV-55.html

The following provides a brief description of the mounted observatory on the modified Boeing plane.

The SOFIA program includes a high altitude airborne observatory consisting of a German-built 2.5-meter (100 inch) infrared telescope mounted in a cavity in the rear fuselage of a highly modified Boeing 747SP jetliner. Scientific instruments mounted in the observatory will be capable of celestial observations ranging from visible light through the sub-millimeter far-infrared spectrum. SOFIA is being developed jointly by NASA and the Deutsches Zentrum fur Luft und Raumfahrt (German Aerospace Center).

Fig. 6. Hierarchical XML-based Tree Structure of a SOFIA Project

When users select a node of information, its respective metadata is displayed on the content page. The metadata includes the following node information: the title, date created, date modified, source information, object description, and file type. The content tab allows modification and creation of nodes on the user-loaded semantic tree structure. Users can also load a different existing tree by clicking on the GUI folder button. Nodes can easily be deleted when selected by cursor, followed by pressing the delete button. The SIS also includes an embedded browser and a web-page parser. When the user visits a certain web-page, s/he is able to utilize the parser for identifying related terms. The current version of the web-page parser is limited in listing several related objects to the user-provided keyword within the web-page.

After entering the keyword in the Search Tab and invoking the "Search" function, the web-site in the embedded browser is parsed for the input terms specified by the user.

The Display Tab returns the output results with words preceding and succeeding the term; providing a context for the result. Instead of looking back through the web-page to determine the context of the keyword, the user can quickly identify the relevance of results on the Results Tab. When users find the content relevant to their query, they can create a new object on the tree via the Scraping Tool or modify content on an existing object.

III. OPENGL SEMANTIC INFORMATION SYSTEM

A. *XML Facilitation*

Throughout the system design in both initial and final versions of the SIS, the XML syntax is facilitated through major system components. This facilitating technology is a communication backbone that propels seamless client-server interaction, Active Directory inter-cluster synchronization, as well as fulfilling metadata representation [6]. In the original design of the Semantic Information System, a 2D tree structure was implemented based on the XML format to represent relationships between objects. The textual and image data of the contents was embedded within the objects tags to take advantage of XMLs inherent hierarchical structure. The original tree display allowed for a convenient, yet purely static and plain interface.

When an object is selected within the tree widget, the metadata is immediately displayed to the user, however the user is able to view only one currently selected object. The information regarding its interrelated components could be only recognized by the objects' text-based title name. The user had to manually browse through the list of objects in order to gather the complete information. This time consuming and visually obstructed process of object identification resulted in a proposal of an "all at once" visualization component design, which accounted for a multiple-object viewing environment, as well as for the ability to effectively browse through the "objectized" hierarchical database. Since 2D display cannot effectively show cross-category relations, the implementation of a 3D tree display to visualize semantic relations was necessary. Rather than to overwhelm users with the sentences of textual content contained in each object description, this proposed design allows user to visually identify whether or not the object of interest is relevant to their search criteria based on the visual tree representation. The tree can be expanded and collapsed while navigating through the available content.

In order to visually represent these interrelated objects, the Open Graphics Library (OpenGL) Application Programming Interface (API) was chosen due for its multi platform capabilities and compatibility with the Qt platform, where initial SIS design was implemented[7].

B. *Initialization Phase*

The implementation of OpenGL features inside the Qt-based environment is achieved via QGLWidget class, specifically designed to provide functionality for rendering OpenGL graphics inside the Qt applications [8]. QGLWidget is a subcomponent of the Qt Vertical Layout widget (QVBoxLayout class).

OpenGL works through a low level programming, and hence allows the programmer to control the rendering of the environment. The programming language allows for what is called texture mapping, technique which is responsible for mapping pixels around primitive polygons.

QGLWidget class is composed of three virtual functions that can be reimplemented in a custom-made subclass: paintGL, resizeGL, and initializeGL.

The objective of initialization function is to set up the OpenGL rendering context and to define display lists. The initialization process involves setting up event filters for supporting mouse tracking and touch-based commands within the QGLWidget. The procedure is also responsible for enabling core OpenGL capabilities, such as texture mapping, shade model, and color/depth settings. The initialization process continues with the configuration of the axis rotation and prepares the extracted XML information for texture generation.

C. *Control Panel*

The traditional GUI represents an effort to display the content on the network but it falls short when it comes to graphical capabilities and user control. The traditional GUI does not offer an instant visual representation of the content.

SIS participants will have an advantage in navigating through SIS database when offered a visual representation of the content, leading to faster decision-making by instantly identifying the necessary information.

PaintGL function contains a sequence of commands that lead to the texture mapping and rendering of graphical objects. The PaintGL function concludes with the buffer-swapping command that refreshes the screen changes via double buffering [9]. Figure 7 shows the 3D Control Panel with features of zoom in/out and rotational options around the X and Y axes. The figure also displays the portion of the loaded SIS project in a hierarchical multi-level structure.

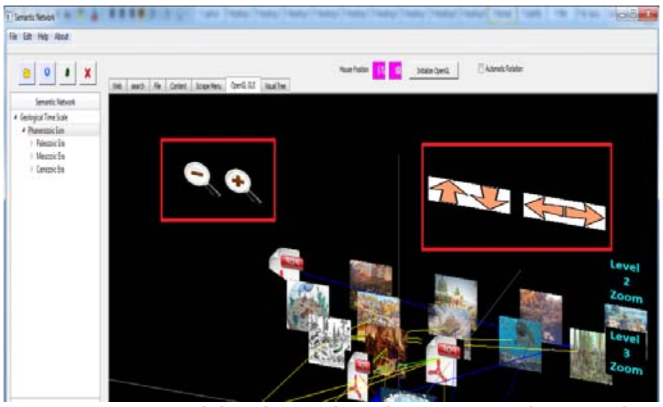


Fig. 7. 3D Touch-based Control Panel GUI (Zoom and Rotational Features)

Current features of the 3D touch-based GUI include regular zoom in/out, level-based filtering (local-zoom) and tree rotation functionalities. The later allows the user to rotate the tree by ten degrees either along the X or Y axes using either the mouse or touch-screen display.

Besides default zoom operations, the right portion of the QGL Widget's rendering region is reserved for what is referred to as a "Local Zoom Menu" (Figure 8). It is designed to conserve visual space on the QGL Widget. Upon user-selection of a necessary level of viewing, the local zoom capability will filter out (visually remove) the distant levels of information from the QGL Widget, while displaying only the level selected and its adjacent levels (parent and child levels).

This feature helps users to visually concentrate on the specific level of information, displaying only the selected level, as well as the nearest preceding and succeeding levels of information. The visual data of all other levels within the tree structure will be temporary concealed from the user.

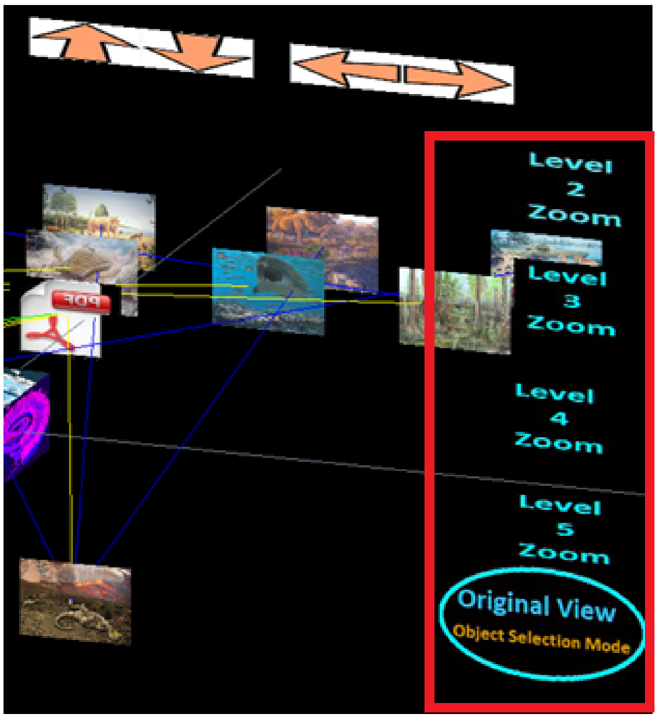


Fig. 8. Local Zoom Menu and Original View Features

D. *Texture Mapping*

The final step in setting up OpenGL GUI environment is the texture mapping [10]. The process initially starts with loading image data of every object into the memory and converting it to the GL recognizable data type. Then, the primitive polygons are created based on the required number of textured objects. Each object is bound onto its corresponding polygon as it is graphically rendered on the widget's reserved region. Figure 9 shows the UML Activity diagram of the texture-binding process. The QTreeWidget class describes an XML tree which contains the interrelated items of a single SIS project [11]. Each of the individual items is represented by the QTreeWidget Item class [12]. The node information is retrieved from the traditional GUI line fields. Based on the extracted metadata information, the corresponding textures are created for graphical rendering.

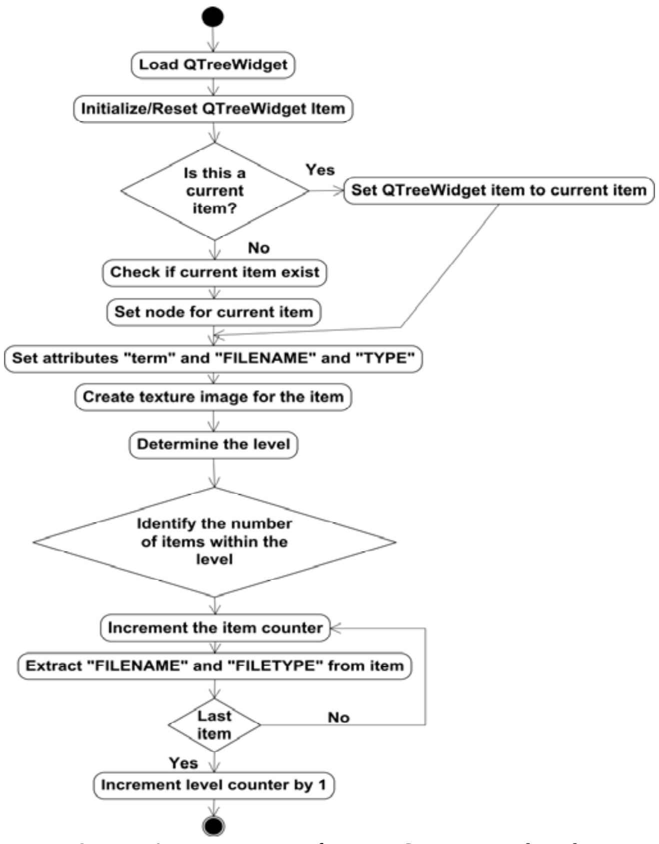
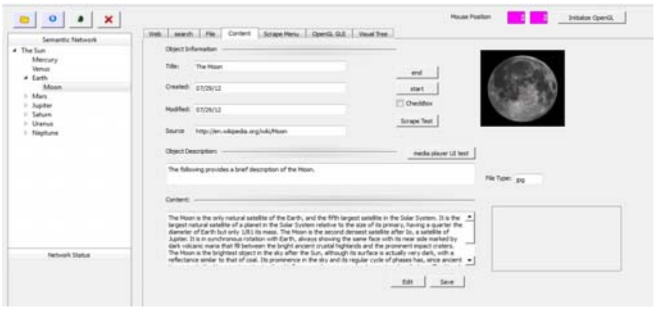

Load QTreeWidget
Initialize/Reset QTreeWidget Item
Is this a current item?
Yes
Set QTreeWidget item to current item
No
Check if current item exist
Set node for current item
Set attributes "term" and "FILENAME" and "TYPE"
Create texture image for the item
Determine the level
Identify the number of items within the level
Increment the item counter
Extract "FILENAME" and "FILETYPE" from item
Last item
No
Yes
Increment level counter by 1

Fig. 9. UML Activity Diagram of Texture Generation and Binding

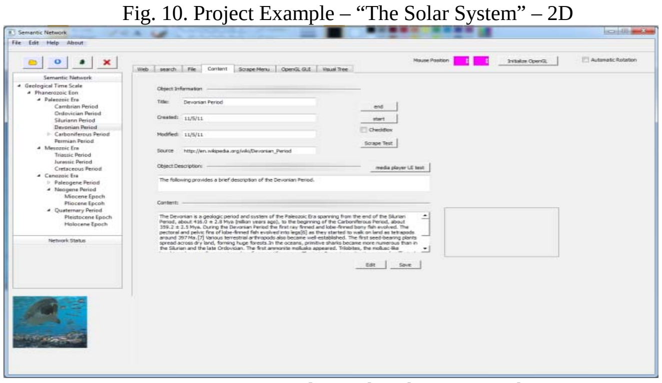
IV. SIS PROJECT EXAMPLES AND IMPLEMENTATION RESULTS

Figures 10 and 11 present two SIS projects inside the original GUI environment. As previously discussed, user is only exposed to a textual content description and a single image representation of the current object within the project. Users are able to navigate the project in a text-based environment embedded in a QTreeWidget display on the left.



Object Information
Title: The Moon
Created: 07/29/12
Modified: 07/29/12
Source: http://en.wikipedia.org/wiki/Moon
Object Description:
The following provides a brief description of the Moon.
File Type: jpg
Content:
The Moon is the only natural satellite of the Earth, and the fifth largest satellite in the Solar System. It is the largest natural satellite of a planet in the Solar System relative to the size of its primary, having a quarter the diameter of Earth but only 1/81 its mass. The Moon is the second densest satellite after Io, a satellite of Jupiter. It is in synchronous rotation with Earth, always showing the same face with its near side marked by dark volcanic maria that fill between the bright ancient crustal highlands and the prominent impact craters. The Moon is the brightest object in the sky after the Sun, although its surface is actually very dark, with a reflectance similar to that of coal. Its prominence in the sky and its regular cycle of phases has, since ancient

Fig. 10. Project Example – "The Solar System" – 2D



Object Information
Title: Devonian Period
Created: 11/5/11
Modified: 11/5/11
Source: http://en.wikipedia.org/wiki/Devonian_Period
Object Description:
The following provides a brief description of the Devonian Period.
Content:
The Devonian is a geologic period and system of the Paleozoic Era spanning from the end of the Silurian Period, about 416.0 ± 2.8 Mya (million years ago), to the beginning of the Carboniferous Period, about 359.2 ± 2.5 Mya. During the Devonian Period the first ray finned and lobe-finned bony fish evolved. The pectoral and pelvic fins of lobe-finned fish evolved into legs[6] as they started to walk on land as tetrapods around 397 Ma.[7] Various terrestrial arthropods also became well-established. The first seed-bearing plants spread across dry land, forming huge forests.In the oceans, primitive sharks became more numerous than in the Silurian and the late Ordovician. The first ammonite mollusks appeared. Trilobites, the mollusc-like

Fig. 11. Project Example – "The Changing Earth" – 2D

Figures 12 and 13 below show the same projects inside of the enhanced SIS 3D touch-based GUI. "The Solar System" project example is shown in three-levels, while "The Changing Earth" project example is shown in five-levels respectively. The amount of levels can be easily identified by the user simply by looking at the visual project structure.

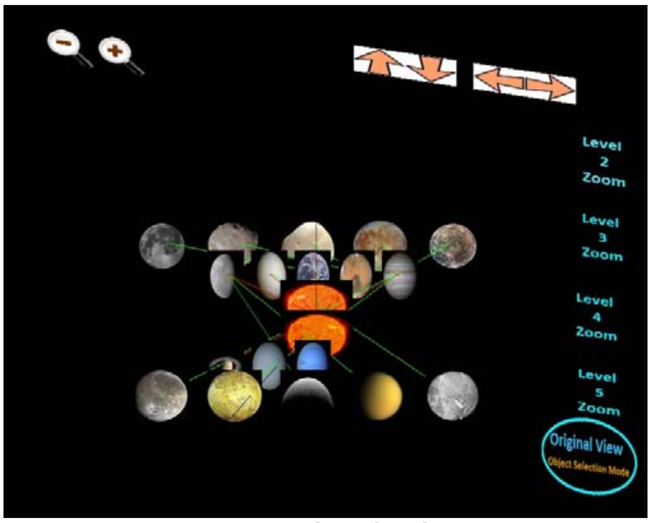


Fig. 12. Project Example – "The Solar System" – 3D

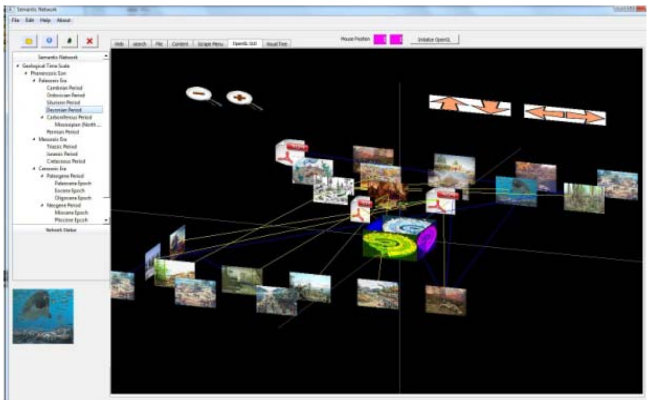


Fig. 13. Project Example – "The Changing Earth" – 3D

Figures 14 shows Local Zoom feature applied on "The Changing Earth" project. The visual information filtering allows the user to focus on the specific level target, when the user selects Level 3 Zoom, the distant level information (levels 1, 2 and 5) is visually removed.

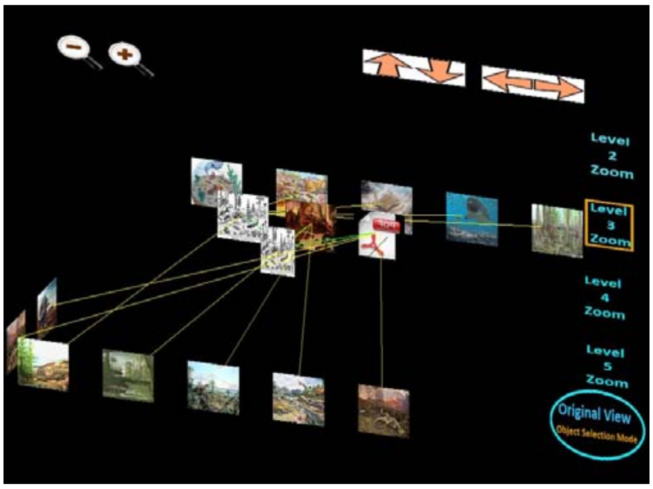


Fig. 14. Local Zoom Feature – Information Filtering

V. CONCLUSIONS

This paper presents the UML design of the traditional and enhanced graphical user interfaces within the Semantic Information System. The SIS core components are thoroughly discussed for both versions of the system.

The 3D GUI interface of the SIS is presented and described from the activity UML diagram perspectives. This interface is also evaluated and compared to the traditional 2D SIS environment. The OpenGL processes of the touch-based GUI are described in details of their operations. The XML parser, the FTP module, and the database system components are presented using UML in respect to the whole system component interactivity.

The comprised system is based on the XML facilitating technology, which allows its platform participants to effectively create, recognize, edit, and manipulate interrelated information in the hierarchical tree-structural format by utilizing visualization environment with flexible touch-based controls. While the traditional user interface is straightforward at storing, retrieving, and modifying the encapsulated content, the lack of the visual aspects hinders the dynamic cross-category visualization.

The flexible control panel with touch-based zoom and rotational capabilities demonstrates visual and navigational superiority in information browsing over static and mostly text-based interface of the earlier version of the SIS.

REFERENCES

[1] E. Tsai, N. Arellano, S. Mendoza, G. Nunez, A. Lin, G. Carter, A. Lam, JP. Adigwu, J. Estrada, A. Milshteyn, H. Boussalis, and C. Liu. *Semantic Information System : Applicaitons in K-12 Education. The Journal of Computing Sciences in Colleges (Vol . 26, Num. 4) , April 2011.*

[2] *W3C Semantic Web Activity* Available at http://www.w3.org/2001/sw/

[3] R. Mathis, L. Caughey *A Metadata Model for Electronic Images* Available at http://mathiswebs.com/papers/mathiscaugheyfinal.pdf

[4] R. Tolksdorf, F. Liebsch, and D. Minh Nguyen, *XMLSpaces.NET: An Extensible Tuplespace as XML Middleware.*

[5] *XML Marker* Available at http://symbolclick.com/navigation.htm

[6] JP. Adigwu, A. Alegre, S. Beltran, J. Estrada, A. Lam, A. Milshteyn, C. Liu, H. Boussalis *Semantic Network Active Directory Service System International Conference on Data Engineering and Internet Technology (DEIT 2011).*

[7] *OpenGL Coding Resources* Available at http://opengl.org/resources

[8] *QGLWidget Class Reference* Available at http://doc.qt.nokia.com/4.7-snapshot/qglwidget.html

[9] *OpenGL Double Buffering* Available at http://swiftless.com/tutorials/opengl/smooth_rotation.html

[10] A. Edward *OpenGL: A Primier 3rd Edition 2007.*

[11] *QTreeWidget Class Reference* Available at http://qt-project.org/doc/qt-4.8/QTreeWidget.html

[12] *QTreeWidget Item Class Reference* Available at http://qt-project.org/doc/qt-4.8/QTreeWidgetItem.html