OPENVIEW
A FREE SYSTEM FOR STEREOSCOPIC REPRESENTATION OF 3D MODELS OR SCENES

L. A. Sechidis[*], D. Gemenetzis[*], S. Sylaiou[*], P. Patias[*], V. Tsioukas[**]

[*] The Aristotle University of Thessaloniki, Department of Cadastre Photogrammetry and Cartography
Univ. Box 473, GR-54006, Thessaloniki, Greece
[lazikas, dgemen, sylaiou @photo.topo.auth.gr], patias@topo.auth.gr
[**] [2]Demokritos University of Thrace, Dept. of Architecture,
New Building of Central Library, GR- 67100, Xanthi, Greece
vtsiouka@arch.duth.gr

SS 4: CIPA

KEYWORDS: 3D representation; visualization; stereoscopic vision; virtual reality; programming; databases

ABSTRACT:

Augmented reality is one of the new applications to archaeology that gives to user the sense of “being there” and allows to observe virtually reconstructed archaeological landscapes with historical buildings.
This paper presents a system that aims on representation of a virtual environment in stereo, using 3 virtual cameras. It allows user to put in scene one or more models that are ready to use and to navigate in them. In addition to this, it gives the opportunity to interact with the virtual environment in real time, to rotate the archaeological findings presented in scene and observe their detail. It establishes links between the objects of the scene and any database and allows user to take additional information about the place or objects.
Since the system is free – anyone can have it with the source code, it can be used from researchers as a tool that will help them on develop and test new techniques on 3d representation (e.g. multiple LOD meshes etc) of any 3d data.
The case study examined in this paper discusses the potentials provided by this system and its advantages and disadvantages.

1. INTRODUCTION

Virtual Reality (VR) and Digital Stereoscopic Systems (DSS) are not a new concept. VR hybrid systems and functional DSS (eg. Photogrammetric Stations) existed even from early 90’s; unfortunately, both had low capabilities and very high cost, due to specialized hardware they needed. At that time, several companies involved to the new VR software industry, with most of their software to considered as a VR world developing tool, while cost varied from some hundreds dollars to as much as lots of thousands of dollars. Additionally, the majority of this software composed of libraries of specific programming routines that were combined to form the substance and dynamic content of the virtual world.

Nowadays, although the same hardware is needed, due to rapid technology progress and the dramatic fall of prices, even individuals can have a fast, powerful and most of all, cheap stereoscopic hardware system. But, the same does not stand for the software. The cost of the software is still the same, if not more expensive, since its capabilities have dramatically increased. For example, the cost a game engine can vary from some thousand dollars to hundreds of thousands of dollars depended on the capabilities and the purpose of its usage.

OpenView comes to fill the gap. Although not a VR developing tool, nor a game engine, is a free of cost VR *presentation* tool. Its purpose is to present any kind of 3d data, from simple points and lines to huge VR scenes with thousands triangles and textures. Additionally, OpenView can be used from researchers as tool that will help on develop and test new techniques on 3d representation (e.g. multiple LOD meshes etc) of any 3d data, since the source code is available, upon request.

2. OPENVIEW IN BRIEF

OpenView can import and handle a virtual world or any kind of 3d data, interact with viewer, render stereo pairs and produce images for both left and right eyes in order to have stereoscopic vision. Also, it can display information or metadata about the objects that participate in the world using any database that is published in ODBC.

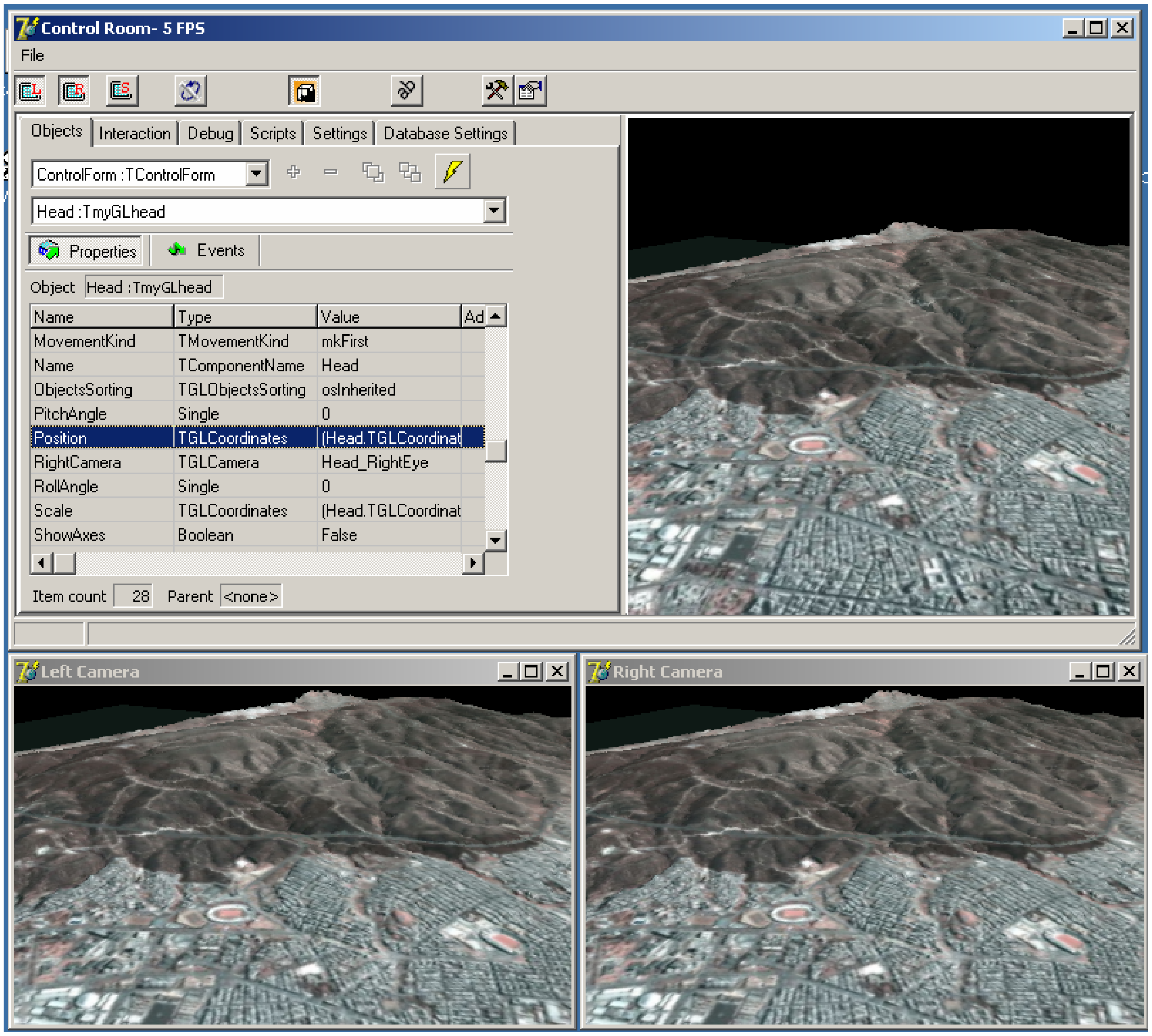
OpenView consists of three major windows. Two of them are OPENGL windows that are projected to viewer, called “left display” and “right display”. The third one, called the “Control Room”, is the heart of the system. Even OpenView can work on a dual-display system, it is recommended to be used with a tri-display capabilities system for better performance and interactivity.

2.1.1 Required hardware

OpenView needs a typical stereoscopic presentation setup: two polarized projectors that will project the stereo pair on a silver-dyed screen. Viewers must wear polarized eye-glasses in order to view the projected stereo pair. For the moment it does not support single-display setups (like DPS do).

Additionally, one or more graphics cards with OPENGL support is needed in order to render the stereo pairs and to output the images to the projectors.

For this implementation and tests, two Compaq projectors having polarized glasses and two systems with different CPU, graphics cards and RAM (both with cost less than 1500 €) were used.



ControlForm :TControlForm

Head :TmyGLhead

Object Head :TmyGLhead

Name	Type	Value	Ad
MovementKind	TMovementKind	mkFirst	
Name	TComponentName	Head	
ObjectsSorting	TGLObjectsSorting	osInherited	
PitchAngle	Single	0	
Position	TGLCoordinates	(Head.TGLCoordinat	
RightCamera	TGLCamera	Head_RightEye	
RollAngle	Single	0	
Scale	TGLCoordinates	(Head.TGLCoordinat	
ShowAxes	Boolean	False	

Item count 28 Parent <none>

Figure 1: Openview in action

3. EXPLORING OPENVIEW

3.1 Importing models and scenes

OpenView is not a graphics editor, nor a VR developing tool. This means that, even it is possible to use OpenView in order to create simple objects and worlds from scratch (using scripts), it is suggested to use external, specialized tools to do this task; then all it is needed is just importing them into OpenView.

3.1.1 Supported formats

Since OpenView uses the GLScene Library, it supports all the formats that GLScene does. Some of them are then 3D Studio (3ds), TIN, STL, MD2 (Quake2, animated), OBJ (WaveFront, and many others), SMD (Half-Life, skeletal animation, obtained from a decompiled MDL, f.i. with MilkShape).

Additionally, an extra implementation has been done in order to import grid data from simple ASCII files or Surfer's DSAA format, both combined with orthoimages.

When a new object is inserted into the scene, it gets a unique and distinguished name. This name is very important, since it is used from the internal scripter to identify the object. It is the parameter that is used to query the database for additional information about the object and it is displayed in objects properties Inspector.

3.2 Viewing and changing objects properties

It is possible, in real time, to view and change almost all the properties of an object (or model). This is done using the Objects Inspector panel in the Control Room, where all objects and their properties are displayed. For example, the position, rotation vectors, scale, colour or visibility of any object can be change. The number of the properties depends of the object's type. The only thing that cannot be changed interactively is the geometry of the object, although this can be done using scripts.

3.3 Moving and rotating

Moving and rotating inside scene is achieved using the PC's keyboard and mouse or joystick. Moving forward and backward, turning and strafing left and right, going up and down (using mouse wheel) has already implemented, while different motion behaviours (e.g. head up/down) are under construction. Motion and rotation speeds are not fixed and can be changed in real-time. Also, motion can be done using two different methods, *walk* or *flight.*

The whole scene or a specific object that participates in scene can interactively be rotated in order to be viewed from several perspectives. A later restoration of the scene to its original position is possible. Rotation is achieved using the mouse.

3.4 Further interaction

Except for the rotation and properties changing, further interaction between viewer and scene can be achieved, with or without scripting. For example, it is possible to select an object, to query a database about the object and display an image, play a video file or a sound that is linked with the object. Also, a script can be run automatically in order to automatically change any properties of the object or even to replace this object with another. The action that will be selected is left to whom will make the presentation.

3.5 Database connectivity

In order to connect OpenView with a database, the database must be "published" to ODBC. OpenView "talks" to database using SQL queries. This way, it is irrelevant if the database is in Oracle, Access or in any other format.

OpenView uses a smart interface that allows the viewer to select the proper table and a **connection field,** from all available databases, tables and fields that are published to ODBC. Also, it allows the viewer to select only the wanted fields from the selected table. This connectivity can be done at any time; additionally, it can switch to another database or table during presentation. For convenience, all database settings can be stored into files and can be loaded when needed.

Every time database info is needed about an object, OpenView creates an SQL query and passes the **name** of the selected object as a parameter to the **connection field** in order to find the specific record from the selected table. This task is done in the background, so it is invisible to viewer. Then the results are displayed to both left and right windows in order to have stereoscopic view of the info.

3.6 Scripting support

Scripting support is OpenView's most powerful component and is based on "Script Studio", from Tmssoftware. The script language that currently supported is "object pascal " while Visual Basic is in the way.

Using scripts, the viewer is able to access every scene component, to change any of its properties, to move or rotate it etc. Actually, it is left on the fantasy to whom prepares the presentation what the script will do.

For example, using the script below, a new object is entered to the scene

```
Axis:=TmyGLAxis.Create(ControlForm); //create the object
Axis.Name:='Axis1'; //Name for scripter and Inspector
ObjectsCube.AddChild(Axis); //put axis to ObjectsCube
Axis.valid:=true; // Enable axis
Axis.RememberMe;  // Set axis visible to scripter
Axis.Position.SetPoint(3,3,3);
```

while the following one (can be as part of the above script or standalone) rotates and moves the already loaded object

```
For i:=0 to 5 do begin
  Axis1.Position.SetPoint(i,i,i);
  AxisCube1.pitchAngle:=i*3;
  sleep(100);
  processMessages;
end;
```

Actually, there are very few things that cannot be done using scripting.

3.7 Producing Stereo Pairs

Stereo pair production is done in a background process, forced to produce pairs as fast as the combination of the CPU and the graphics card allows. OpenView uses three cameras when running. The first (central camera) displays its contents on the Control Room. The other two (called "left eye" and "right eye") are responsible to create the stereo pairs of images. All three cameras together are called the "Head". Rotation of the "head" rotates all three cameras at once. The position of the two cameras is not fixed but floated. Even the two cameras lie on the same line that passes from the center of the central camera, their distance can be change. This is done because, most of the times, different scenes need different "eye distances".

There are a couple ways of setting the virtual cameras and rendering the stereo pairs: toe-in and off-axis projections (Burke, 1999). The toe-in projection (Fig. 2b) is easier to be implemented (just set the two virtual cameras focus at the same point) but has one major disadvantage: creates stressful stereo pairs due to vertical parallax. The off-axis projection is more difficult to be implemented since it requires a non symmetric frustum which is not supported by all rendering packages but produces less stressful stereo pairs (fig. 2a). By default, OpenView uses the off-axis projection. But it is also possible to use the toe-in projection, too.

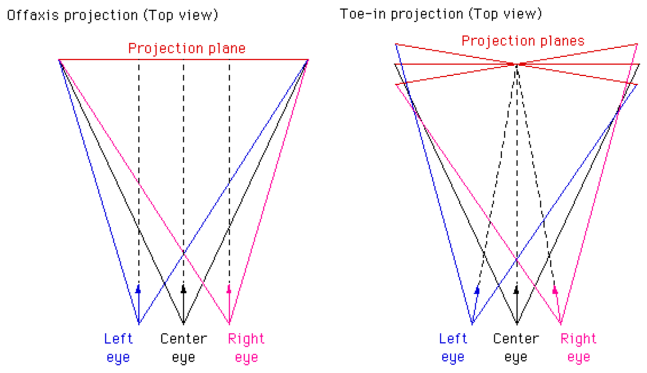


Figure 2: Off-Axis (correct) and Toe-in (incorrect) projections

3.8 Creating 3D videos

OpenView can be used in order to create 3d videos of the object or the scene. All OpenView's "head" cameras can save, on demand, single images or avi files of the rendered scene. Using this ability, creation of 3d videos is an easy task: OpenView is used to produce left and right images of the scene and saves them as image or avi files. Then an external application (3D Video Creator) imports them and produces the video.

Additionally, the position and the rotation of every camera can be saved into a file and then stored as metadata to the video stream, in order to create geo-referenced 3d videos (Sechidis et al, 2001).

4. PERFORMANCE TESTS

In order to test OpenView's performance, several tests have been made, using two different CPU and graphics card combination setups and different virtual worlds. Next table shows the results of these tests, while additional details about tests follow:

	Speed (fps) for each display	
	Setup 1	*Setup 2*
Loading test 1	20 (sec)	7 (sec)
Test 1	3	6
Test 2	150	200
Test 3	130	170
Test 4	20-30	55-60

Table1: Performance results for several tests

Setup 1: Pentium 4 1.8 GHz, 512M RAM, Matrox P750, 64Mb RAM
Setup 2: Pentium 4 2.8 GHz, 1G RAM, Nvidia 5600 XT, 256 Mb RAM

Test1: One surface (453 x 526 points grid) with a 3392 x 3935 pixels, RGB ortho image of Thessaloniki (as shown in Fig. 1).

Test 2: An archaeological site comprised of about one thousand polygons, using texture materials.

Test 3: The site of the test2, loaded 10 times in different positions

Test 4: A 3ds file with 67392 vertices (58928 faces) having 1214 objects and 15 different textures.

Both left and right displays were 1024 x 768 pixels wide, while Control room was about 400 x 300 pixels wide in all tests.

As shown above, OpenView's performance was good enough for most of the tests, except the first one. In that test, the one big surface is responsible for low performance since it had to always be rendered, no matter of the viewer's position and view angle. The solution to this (which is under construction) is to split the one big surface to a lot of smaller ones.

5. CONCLUSIONS – FURTHER WORK

OpenView is a generic tool for stereoscopic representations of 3D objects or VR scenes. It is common sense that a specialized tool for presentations of archaeological sites has different requirements from an equivalent of medical applications or from a tool that will present VR worlds for entertainment. However, OpenView can be used with success in all above applications even if it has the minimal requirements that these applications require.

Performance tests have shown that it can be used for real time presentations, even with the usage of low cost CPU systems.

The further growth of OpenView will continue be focused in the implementation of capabilities that will have generic character. Thus, in future versions they will be added animation, 3D (or 5.1) sound, internet abilities (client - server), ability of creating scenes using two computers, possibility of measurements and motion that will be based on physics (ODE). For more specialized requirements, the team of OpenView hopes in the help of its users. Since the source code will be available, is given the occasion in each one to face and to resolve each own problem. All the new solutions will be incorporated in the OpenView so as to it becomes more powerful, useful and functional.

Additionally OpenView, coded in Delphi and using public domain or shareware components, depends on GLScene library, an active open-source project; therefore any improvement on GLScene or other component performance will improve OpenView, too.

References

Ogleby, C., 1996, A reconstruction of the ancient city of Ayutthaya using modern Photogrammetric techniques, IAPRS, Vol. XXXI, Part B5, Com. V, Vienna, 1996, pp. 416-425.

Ogleby, C., 2001a, "The ancient city of Ayutthaya-Explorations in virtual reality and multi-media", Proc. of International

Workshop on Recreating the Past -Visualization and Animation of Cultural Heritage, Ayutthaya, Thailand, 2001.

Ogleby, C., 2001b, "Olympia: Home of the ancient and modern Olympic Games. A VR 3D experience", Proc. of International Workshop on Recreating the Past -Visualization and Animation of Cultural Heritage, Ayutthaya, Thailand, 2001.

Sabry F. El-Hakim, L. Gonzo, M. Picard, S. Girardi, A. Simoni, E. Paquet, H. Victor, C. Brenner, 2003, "Visualization of Highly Textured Surfaces", 4th International Symposium on Virtual Reality Archaeology and Intelligent Cultural Heritage

Sechidis, L, V. Tsioukas, P. Patias, 2001, "Geo-referenced 3D Video as visualization and measurement tool for Cultural Heritage**",** International Archives of CIPA, vol. XVIII-2001, ISSN 0256-1840, pp. 293-299

References from the Web

Burke,P, 1999, Calculating Stereo Pairs, http://astronomy.swin.edu.au/~pbourke/stereographics/stereorender/ (accessed April 23, 2004)

Lischke, M, Eric Grange, GLScene: OpenGL Library for Delphi, http://www.glscene.org (accessed April 23, 2004)

scriptStudio,
http://www.tmssoftware.com/ (accessed April 23, 2004)

Authors would like to thank Prof. Cliff Ogleby for his masterful comments and suggestions about this project.

* L. Sechidis has granted a Post Doctorate scholarship from the State Scholarships Foundation of Greece.