USING GAME ENGINES FOR NON 3D GAMING APPLICATIONS

Jerome Dupire, Alexandre Topol, Pierre Cubaud
CEDRIC/CNAM
292, rue Saint Martin 75003
Paris, France
E-mail: {dupire, topol, cubaud}@cnam.fr

KEYWORDS
Game engine, 3D application, Digital library, Virtual reality.

ABSTRACT
We present in this paper the use of game technologies during the development of 3 dimensional interfaces for accessing digital libraries. We compare these different tools through a practical point of view and propose to consider them in a more general context of 3D applications development.

INTRODUCTION
At CNAM, two digital libraries (DLs) born from our laboratory are accessible. The *Association des Bibliophiles Universels* (ABU, http://abu.cnam.fr), started in October 1993 and has today thousands of books downloaded daily and has become one of the most active French speaking cultural Web sites. The *Conservatoire Numérique* (CNUM, http://cnum.cnam.fr), was put online in January 2000. Three hundred reproductions of old scientific and technical books are accessible on this Web site. These digital libraries are accessible online with a WWW-based interface and a standard architecture.

We present in this paper different projects for accessing online DL and fulfil the catalogue browsing and the reading tasks. Since 1998, our interfaces have evolved in design and interactions, but have also taken into account different technical choices. We will explain how our prototypes have evolved, from the original, VRML based, one to the last ones, which were built with modern video games APIs (*Application Programming Interface*).

3D DIGITAL LIBRARIES INTERFACES

Collection Browsing
We have tried to provide to CNUM users and librarians a similar experience using 3D visualization (Cubaud et al. 1998). We collected books bindings images and arranged them into a 3D scene depicting a virtual, very large, shelf. In such a scene, the user can infinitely rotate the point of view and move in order to zoom in or out.

Binding Browsing and Reading Tasks
After a first attempt to design reading tools using 2D transparent windows (Cubaud and Topol 2001) in a VRML 3D scene, we developed a new metaphor of book (the tripod on Fig 1), which allows the reader to handle and work (e.g. read) on digital documents. The POV (*Persistence Of Vision)* scene on Fig 1 depict a "cockpit" architecture, similar in spirit to the Web Forager interface (Card et al. 1996). Reading books and browsing collection can be done in the same space

Fig 1: The whole collection is rendered and accessible in the background. Tripods allow to move and read facsimiles. Favorite books are shown in the foreground.

3D IMPLEMENTATION PROBLEMS
Each new generation of dedicated hardware for real-time 3D exhibits a considerable growth of processing speed, storage capabilities and, to less extent, screen resolution. Experiments described in (Cubaud et al. 1998) were in 1997-8 below acceptable frame refresh rates for fluid interaction (about 25 frames/s is considered in the game industry). The memory dedicated to textures management was at that time too small for scenes such as the one reproduced on Fig 1. It is today quite possible to handle scenes that hold about 10K different books textures (which is the forecasted size of the CNUM collection). Pages are bigger image files, and at most a hundred can be stored directly in the graphic card's memory. This forbids "brute force" algorithms for fast page turning with large books. This problem has recently been studied in (Card et al. 2004).

The prototype described in (Cubaud and Topol 2001) was mainly written in Java and relied on the 3D engine enclosed in a VRML browser. The amount of scripts required to implement the interaction behaviors of the 3D objects is important : over 90% of the demonstrator's code is made of Java scripts. Telling that VRML and Java is a useless couple to create interactive scenes is not the point. We do believe

that almost every imaginable interaction can be coded this way. But it demands a long coding and debugging time. To ease the specification of such interactive scenes some extensions to VRML were proposed in (Topol 2002) along with a description of the corresponding client navigator. Nevertheless, the VRML language, like a game scripting language, is based on an specific engine (the VRML plugin) which only allows to create applications with predefined appearance and interactions (those enclosed in the VRML engine). Hence, creating a tailored 3D application is impossible. For instance, with the Quake engine, one can only build FPS-like applications or games.

The prototype based on the book metaphor is a stand-alone application written in C++ and taking advantage of the OpenGL library (Cubaud et al. 2002). All 3D objects were procedurally coded, as well as the texture mapping. The demonstrator includes a simplified collection shelf, tripods creation/deletion and page turning functions. This software has been demonstrated to IST professionals at JFT'2003 (Cubaud et al. 2003) and ECDL'2003 (Dupire 2003). A wider audience has been reached during a two-weeks long demo session for the annual "Image par image" exhibition (Montreuil, France) in march 2003. In October 2004, a similar demonstration took place at the CNAM library for the "Science en fête" nation-wide event. Each time, a dedicated system was installed for the 3D environment, while another offered access to the CNUM website. A time switch limited the session to a few minutes and user's actions were logged. We have concluded from these demonstrations that users reached quite quickly an understanding of the tripod operations. The page turning functions have been acclaimed by the general public and librarians (although not always by computer professionals). Observation of users droved us to modify the interface. The moving distance for tripods has been bounded: a tripod can not be bigger than the screen, nor too small when in the background. Collision detection has been improved to detect the user's actions in the scene. This was done by ray casting from the 3D cursor. Scene lighting and objects shadows have been implemented to ease the positioning task of the tripods. If the solutions to improve the interface usefulness were evident, their implementation was heavy to do and time consuming (i.e. the ray casting had to be entirely hard coded).

GAME ENGINES SOLUTIONS

Like in almost every 3D applications, our interface needs a good representation, animation and interaction management. This means :

- a tight connection with a 3D modeler,
- simple to use animation procedures,
- and build in input devices functions.

3D APIs like OpenGL were created for CAD software rendering and do not provide many other functions. DirectX contributions are still very low level and demand important programming skills to achieve simple interactions. With both APIs, no model loaders, collision detections or animation manager is given. On the contrary, the world of video games was confronted with this type of constraints since a very long time. With the apparition of the first 3D games and thanks to the increasing capabilities of the hardware, programmers found many solutions in order to unceasingly improve the quality of graphics, the fluidity of animations, the immersion with sound environments. It is far beyond the simple world of video games that this progress is reflected. No doubt that the breath generated by this medium is a great part of the evolution of graphics cards and more generally the whole computer configurations. For example, one can see that new functions are available on each new graphic cards, that processors have new opcodes (SSE, SSE2, 3D Now!) for working on matrices, that motherboards evolve to allow the important data flow necessary for graphics cards (AGP and PCI Express buses).

Among the many technical contributions generated by this evolution, the "game engine" is one of the most interesting. It encloses many aspects that need to be managed within a game. Since one wishes to obtain the best graphics and to reach many programming functionalities, these game engines become impossible to circumvent. Those can be commercial products or private engines developed by game studios in order to be used for several projects. With such engines it becomes easier to program a 3D game and more generally a 3D application. The first and most famous is the Quake engine (IdSoftware), available as an open source format. It allows to create Quake-like games or applications. Since then, game engines have evolved toward complete software architectures for games. They became supple enough to permit the programming of almost any 3D game.

Developed by and for the programmers, these game engines offer an environment created especially to implement the functionalities that are specific to the 3D video games. Thus, the management of an animated model, the collisions between objects, the interaction between the player and the game, through the keyboard, the joystick or the mouse, are as many aspects whose implementation is facilitated. And time won with these aspects can be consumed to think and implement the right interactions (known as the gameplay for games). The most recent games use engines that go beyond visual and interaction aspects. For example, programmers can rely on :

- a physical engine to simulate physical laws within the virtual environment,
- an audio engine to add music and complex acoustical effects,
- an artificial intelligence (AI) engine to program non human players' behaviors.

Properties awaited for a 3D interface are very close to the ones for 3D video games. Indeed, when creating a 3D interface for information or document visualization, the rendering and animations must be optimized. Like in games, "visual heuristics" are also one example of the "many complex information tasks [that] can be simplified by offloading complex cognitive tasks onto the human perceptual systems" (Hearst et al. 1996). From a gamer's point of view, one can say that frame per seconds is one of the most important thing when dealing with a 3D interface. Taking this fact into account when programming with a game engines is easier than with standard APIs. Some engines can guarantee a minimum frame rate and most give tools to tune an application. For this reason, game engines

represent a good software architecture for implementing any kind of 3D applications and not only games.

The easiest and less time consuming way to implement a DL with a game engine would have been to use a quake-like engine. Christoffel and Schmitt chose this original solution (Christoffel and Schmitt 2002). With such scripted engines, one can only create games or applications that look like a shoot game with player or user enclosed in a maze. Only a real looking like library can be recreated and this solution does not meet our design choices. We think that in a real or "virtually real" library, unneeded moves are often necessary to find the shelves containing the books of interest, to pick it and to go back to a reading desk to work on it. These steps can be simplified in a virtual DL to ease the training of inexperienced or occasional users. Hence, to allow this, we have used 3D game engines.

We have developed two prototypes of digital libraries using game engines. The first one uses the Virtools engine (Fig 2), initially conceived to create on-line web games. It seemed to be very practical because one can easily describe any kind of application. To achieve that, Virtools is operating through two main parts : the first one, Virtools Dev makes it possible to specify how the application works, thanks to a graphic programming system (behavior boxes and connecting arrows). The second one is a plugin for web browser which allow to download and play the application. This environment is easy to use and to provide to the final user.
However, its graphical programming environment proved to be unpractical for the implementation of interactive 3D objects such as ours. Indeed, building new boxes (i.e. when a behavior is needed and not available in the software) is quite tough and complex. An other point of view is that Virtools, with it's HTML compliant browser, is mainly dedicated to web applications design and does not fit to stand alone applications.

Fig 2: A simple Virtools prototype.

Then, the second solution was a complete rewriting of the OpenGL software, using Criterion's Renderware Graphics (RWG). A screenshot of this prototype can be seen in Fig 3. RWG is a mid level API (middleware), lower than Virtools. RWG functionalities are integrated into C or C++ programs through the use of libraries (.lib) and headers (.h). It means that RWG provides to the programmer a lot of functions allowing a higher management level of graphics than OpenGL do. For instance, the 3D scene rendering is achieved with only one function (*RpWorldRender()*). By the way, and for the main aspects of an interactive software development, the programmer can easily (and rapidly) design each part of it.

The collision detection is widely used to manage the user's actions. Bounding boxes are computed for each part of interest of the 3D objects in the scene (i.e. pages, tripod's root, books on the shelves, …). The collision detection test is launch in every frame, called by a high level function (*RpAtomicForAllIntersections()*). Most of things are already coded and included into this function. The programmer only needs to write a callback function to specify a reaction to a collision.

On the other side, since RWG is build over OpenGL or Direct3D (two distinct API are provided), the programmer can tune the code with his/her own functions. RWG go further than OpenGL by giving tools, not only for the 3D graphics management, but also for the main aspects of any 3D interactive software (i.e. collision detection, picking, 3D objects animation, 2D graphics management, Macromedia Flash binding, etc...). Thus, our experience with RWG was interesting, even if the prerequisite programming skill was more important than with Virtools. Another interesting aspect of RWG is that it is platform independent, i.e. the same code can be compiled for PC or any kind of other entertainment system (PS2, Xbox, etc..).

A main drawback of RWG is the slow integration of new rendering techniques. Since RWG relies either on OpenGL or Direct3D, technical improvements of the rendering pipeline must first be integrated within the low level APIs. Another issue is the cost of such professional game engine that can only afforded by rich companies.

Fig 3: Textures, 3D objects and interaction can be easily handled in a RWG based application.

Finally, both of the engines were compliant with most famous 3D modeller (i.e. 3DSMax, Maya). This was the origin of a important time gain, dealing with the 3D objects building step. Importing 3D object in the scene was quite easy and direct. This new feature (compared to the older technologies) allows more flexibility, even creativity in the

design process of the objects, by using a dedicated tool to do it (versus a procedural "design").

CONCLUSION – FUTURE WORK

We presented in this paper a 3D digital library development project, using different programming technologies, from VRML to the up to date game engines. Benefits of these last ones are obvious, providing on one hand, high level function to manage usual interaction aspects and, on the other hand, allowing the programmer to tune the application regarding to his/her needs. However, these solutions are very expensive because RWG and Virtools are commercial products, initially aimed to the video game industry.

The following table summarizes the pros and cons of each studied solutions based on our experience. To each criterion evaluated is given a mark from 1 (bad) to 5 (excellent).

	Description (VRML)	Visual prog (Virtools)	3D API prog (OGL, D3D)	3D GE prog (RWG)
Rendering				
Speed	3	3	5	5
Quality	1	3	5	4
Flexibility	1	2	5	3
Animation & Interactions				
Collision detection	2	4	1	5
3D interaction	2	3	2	4
Animations	3	3	1	5
Resources				
File loading	3	4	1	4
Documentation	5	2	5	3
Examples	5	2	5	5

Our next aim is to provide a better immersion feeling to the reader by using a semi-spherical Elumens 'Vision Station' (VS) display. Sold with it's own API for correcting image projection, the final integration is not as simple as we first expected. Indeed, the VS API has to be handled in an OpenGL context. So we need to access to the OpenGL lower functions to use it. Thus, we are going to study and compare the open source 3D game engines, in order to have both access to low and high programming level. A lot of free 3D engines already exist, with various features. Our first step will be to compare the Irrlicht engine (Irrlicht) with the Delta3D one (Delta3D). Both are free and open source and the Delta3D user community have already developed a add-on library to connect with the VS. By the way, we should have the capability to build easily complex 3D interactive environments, composed with heterogeneous devices like immersive display, position sensors or datagloves.

REFERENCES

Card S. K., Robertson G., York W. "The WebBook and the Web Forager : An Information Workspace for the World-Wide-Web", *In Proc. of ACM CHI'96*. Vancouver, Canada, April 1996.

Card S. K., Hong L., Mackinlay J. D., Chi E. H. "3Book: A Scalable 3D Virtual Book", *In Proc. of ACM CHI'04*. Vienna, Austria, April 2004.

Christoffel M., Schmitt B. "Accessing Libraries as Easy as a Game", *in ACM JCDL Workshop*, USA, Portland, 2002.
Cim A. *Le livre*. Flammarion, vol. 5, p. 218, 1990.

Cubaud P., Thiria C., Topol A. "Experimenting a 3D Interface for the access to a Digital Library", *In Proc of ACM DL'98. Pittsburg*, USA, June 1998.

Cubaud P., Topol A. "A VRML-based user interface for an online digitized antiquarian collection". *Proc of ACM SIGGRAPH Symposium Web3D'2001*, Paderborn, Germany, Feb. 2001.

Cubaud P., Stokowski P., Topol A. "Mixing Browsing and Reading Activities in a 3D Digitized Library". *In Proc. of ACM-IEEE JCDL'02*, Portland, USA, June 2002.

Cubaud P., Dupire J., Topol A. "Textes, images, volumes : les bibliothèques numériques au CNAM". *Premières journées francophones de la toile (JFT)*. Tours, France, july 2003.

Delta3D Engine at *http://delta3d.org/*

Dupire J. "Digital libraries at CNAM : 1993-2003". *In Proc. of ECDL'03*. Trondheim, Norway, Aug. 2003.

Hearst M., Kopec G., Brotsky D. "Paper and Digital Documents", *in D-Lib Magazine*, ISSN 082-9873, June, 1996.

Irrlicht Engine at *http://irrlicht.sourceforge.net/*.

Topol A. "Interaction 3D pour les paysages informationnels", Conservatoire nat. des Arts et métiers PhD thesis, 2002.