The Mental Vision Framework - A Platform for Teaching, Practicing and Researching with Computer Graphics and Virtual Reality

Achille Peternier, Frederic Vexo, and Daniel Thalmann

Virtual Reality Laboratory (VRLab)
Ecole Polytechnique Federale de Lausanne (EPFL)
{achille.peternier,frederic.vexo,daniel.thalmann}@epfl.ch

Abstract. Despite the wide amount of computer graphics frameworks and solutions available, it is still difficult to find a perfect one fitting at the same time many constraints, like pedagogical intents and user-friendliness or speed with high rendering quality and portability. In this article we describe our contribution to the topic: the Mental Vision platform. Mental Vision is a framework composed of a teaching/research oriented graphics engine simplifying the users needs in computer visualization and a set of corollary tools specifically designed for practicing and learning of computer graphics and virtual reality. In this dissertation we explain our approach design and the contribution brought into a series of study cases to show how concretely Mental Vision satisfies existing needs not addressed by other solutions.

Keywords: Computer Graphics, Virtual Reality, Teaching and learning, CAVE, Mobile devices, Immersion.

1 Introduction

Virtual Reality (VR) is a science that has gained an increasing amount of popularity and applications during the last years. This increasing interest has also produced a very wide amount of both software and hardware technologies to support and improve creation of VR environments. Unfortunately, most of these innovations are often accessible only by skilled users with a good background knowledge in VR and Computer Graphics (CG) and through cumbersome and expensive devices. VR applications require also a significant amount of time to be developed, because of the complexity introduced by the generation and adaptation of 3D objects to fit into a specific realtime software. Finally, VR is a complex matter by itself and difficult to learn because of the points previously cited and also because of the heterogeneity of notions (mathematics, networking, physics, etc.) a user needs to practice with before attempting to implement a full Virtual Environment (VE).

With the Mental Vision project we aim to tackle theses issues, by creating a framework for virtual reality intended to be very intuitive, powerful and low-cost.

Mental Vision has been created by mainly targeting to the needs and constraints of the educational and scientific community, notably reducing the learning curve and time required to create VEs, limiting the cost of immersive or wearable frameworks without compromising their quality, and improving understanding of VR concepts and techniques by directly practicing with them. Moreover, such a framework has been designed to fit into a wide range of heterogeneous devices, spacing from low-end student PCs through mobile devices up to Cave Automatic Virtual Environments (CAVEs) [1], across different operative systems and hardware setups.

Our framework is composed by two main entities: a multiplatform, multidevice 3D graphics engine called MVisio (see 3.1) and a set of tools ranging from pedagogical interactive demonstrators to 3D model exporters and file format converters (see 3.2).

We introduced our first results about the Mental Vision platform in [2]. In [3] we separately described the mobile aspects of the platform, while in [4] we published the first adaptation of our system to work in a CAVE. In this dissertation we insisted in making complex things easier and more affordable for the users, in order to widen the fruition and application scenarios of VR-oriented technologies. Our experience with students and researchers has shown that this approach effectively responds to a need that is not only welcome but also required. In this work we expose in detail each step of the creation and design of the Mental Vision framework, comparing our approach against other similar scenarios, and we conclude with a series of case-of-study of concrete utilizations and applications developed through the use of our framework *in toto*, pointing out the contributions brought.

2 Related Work

A wide effort has been invested during the last years into the production of tools to standardize and simplify access to 3D visual contents. Industry, researchers and the open-source community released a large amount of platforms to fit the different needs, covering almost each possible device and operating system.

2.1 Open and Closed Source 3D Graphics Engines

Ogre[1], Crystal Space[2] and Irrlicht[3] are among the most used and complete open-source graphics engines available so far that could be used as foundation for virtual reality applications. Similar to MVisio, they feature high quality rendering but suffer from being available only on PC, without support for mobile devices or CAVEs. There are also many differences between the application programming interface (API): such open-source engines are mainly oriented to performance and game development, while MVisio aims at simplicity and compactness.

Industry graphics engine are state-of-the-art frameworks featuring the most recent effects available and extremely optimized for speed. Among these systems

[1] http://www.ogre3d.org
[2] http://www.crystalspace3d.org
[3] http://irrlicht.sourceforge.net

there are Cryengine2 from Crytek[4], Unreal Engine 3 from Unreal Technology[5], Half Life 2 from Valve Software[6] or Doom 3 from ID Software[7]. All these closed-source software are extremely powerful engines but limited by their licence costs, the lack of a full access to the source code for adaptations and modifications required to move the system to other platforms/devices not originally aimed by their developers (like CAVEs or Smartphones) and are very difficult to program for not specialists. Such engines are also very complicated and poorly documented, requiring a good amount of time to get used with as pointed out by Kot et al. in [5]. Kot et al. also complained about the limitations and bugs of some of the GUI available in these engines: with MVisio we implemented our own integrated GUI, offering the same basic user interface functionalities independently from the system/device the engine is running on.

Virtools[8] is a professional framework conceptually very similar to our one, offering support for a wide range of devices and platforms and includes a visual editor for quickly developing applications. With MVisio we preferred to directly simplify programming without recurring to an *ad hoc* visual editing tool and by targeting mobile devices, not supported by their platform. Virtools is also more hardware demanding than MVisio, which has been programmed to work on older student machines too.

Recently Microsoft released XNA 2[9] which is a free development framework oriented to game developers. XNA aims at reducing the complexity of this task by offering all the basic tools and source code to immediately start working on the application itself instead on corollary tasks. XNA is unfortunately very oriented to videogames and limited to Microsoft systems (Windows or XBox consoles) and lacks support for VR and mobile devices. XNA is also accessible only through C# and the .NET framework, a limitation for our students and researchers using also Linux or MacOS systems.

Java3D is acquiring more and more interest thanks to the support for hardware acceleration and availability of the J2ME platform on several recent devices. We used C++ and Windows Mobile instead because at the beginning of our project hardware acceleration on mobile devices was not as robust and as accessible as today. This evolution is summarized by Soh et al. in [6].

2.2 Educational Frameworks

Towle et al. first identified in 1978 with GAIN [7] the interest in offering interactive applications to use at home as an option to practical work sessions, done in cooperation with other students, and as a more intuitive way to learn than with just a manual or a workbook. We extended this idea with our pedagogical demonstrators, by integrating them directly with the class documentation and

[4] `http://www.crytek.com`
[5] `http://www.unrealtechnology.com`
[6] `http://www.valvesoftware.com`
[7] `http://www.idsoftware.com`
[8] `http://www.virtools.com`
[9] `http://www.xna.com`

by making them mobile for at-desk support during practical sessions. The benefits offered by multimedia contents for CG teaching purposes are also shown by Song and al. in [8]: interactive modules reduce learning time and improve use and diffusion of contents over the web, potentially targeting more people without additional costs. Meeker used a learning by practice approach in [9] through the use of the free software Anim8or[10] to let students practice about basic notions of 3D modeling during the course. Our system is more oriented to the use of models already created by artists, for example by exporting them through our plugin from 3D Studio Max directly into the graphics engine.

Many CG and VR topics can also be taught by recurring to games: for example Hill and al. used in [10] puzzles and games to reinforce the learning objectives. Similarly, Becker in [11] used video-games as motivator for programming applications on Computer Science classes. We already had a positive feedback by offering gaming projects during the course of advanced Virtual Reality in [12]: with MVisio, we want to provide to the learners a tool allowing them to benefit more from this option, by giving them all the instruments they need in an easy and comfortable way. Korte et al. also reported that creating videogames may also be an innovative method for teaching modeling skills in theoretical computer science [13].

About the creation of pedagogical oriented graphic engines, when 3D graphics accelerator cards for micro computer weren't available, Clevenger and al. developed a graphics engine to supply students with a learning platform (called TUGS) in [14]. Their goal was to offer a support to students to immediately render some images and to allow them to substitute parts of code of the TUGS engine with their own later in the semester, in order to have a full working platform since the beginning of the class. Their approach was particulary useful before the large introduction on personal computers of graphics APIs based on 3D hardware acceleration like OpenGL and DirectX, which substituted the expensive need to develop a custom rasterizer. Coleman et al. created Gedi [15], an open-source game engine for teaching videogame design and programming in C++. Based on the same principle to create a pedagogical engine, with MVisio, we want to provide a more generic software that can be used not only for games but also for Computer Graphics and Virtual Reality applications, by adding for example support for VR devices like head-mounted displays or CAVE systems. Tori et al. used Java 3D, small videogames, and customized software in [16] to introduce CG to students. They also relied on a more complex Java 3D graphics engine (called enJine) for the development of semester projects. The main advantage of their approach is the operativing system independency offered by Java, very useful when addressing a wide user audience like a student class, using different PCs and operative systems. With our system we more aim at Virtual Reality and supporting devices other than standard personal computers.

In [17], Wilkens pointed out the advantages and disadvantages of using more than one API to access graphics functionalities during a computer graphics course, resulting in a excessive burdening of both students and teachers demanding more extra-time than the time normally expected for the class.

[10] `http://www.anim8or.com`

In the next sessions we describe first the goals and design of our framework and we analyze then different concrete uses of our platform on different projects and classes, to evaluate the contribution brought by our approach.

3 Mental Vision Platform

Mental Vision is a computer graphics and virtual reality framework oriented towards education and scientific research needs. We decided to create our own system after comparing the different already existing solutions without finding a perfect one fitting at the same time to all our constraints. We can enumerate our requests in ten points:

1. Multiplatform (across different operative systems) and multidevice (running on handheld devices, PCs and CAVE environments).
2. Very compact in sizes and resources, reducing external dependencies, improving compatibility between recent and older machines and project deployability.
3. Extremely simple to use, reducing the learning curve and the lines of code to write to achieve results and getting rid of all the corollary aspects CG usually requires.
4. Featuring an embedded GUI system with basic windowing, text management, buttons, etc. without requiring external additional dependencies.
5. Robust, for uses during public demonstrations, classes and conferences.
6. Not dependant from costly software or hardware, being aimed to education and science, often very budget-limited.
7. Fast and modern, featuring a good rendering speed, quality, and satisfying at the same time teaching and research needs in our field.
8. Virtual-Reality aware, easily supporting VR specific devices like Head-Mounted displays or haptic devices.
9. Including the necessary tools to import models, textures, videos and animations from other programs.
10. Including tutorials and demonstrators about the framework itself, CG and VR concepts as well.

With our framework we address and propose a solution to these points. Our work is divided into two main entities: the 3D graphics engine itself (called MVisio) and the pedagogical tools (like modules and tutorials) that rely on the top of MVisio. We describe them both in the next subsections.

3.1 MVisio 3D Graphics Engine

The MVisio 3D graphics engine is a generic, multi-platform and multi-device library giving access to modern computer graphics via a very simple API (see figure 1).

We insisted into simplifying things usually complex like advanced shading techniques or CAVE/mobile devices porting by making them almost invisible

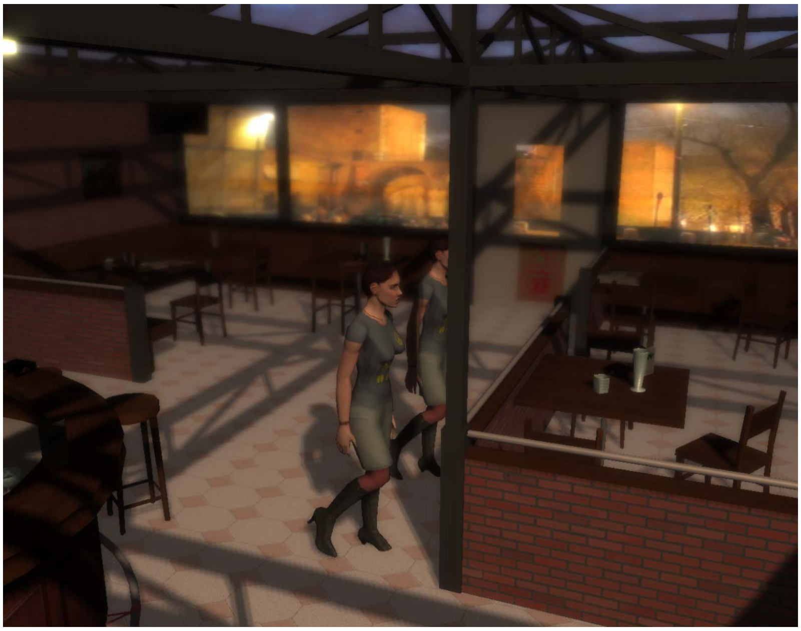


Fig. 1. High-quality rendering with dynamic soft-shadowing, depth of field and bloom lightning

for the user. We extended this idea of simplicity to each aspect of the engine design, aiming at the same time to an apprentice user and a more experienced one. New users can have immediate results with just few lines of code: initializing the engine, loading a scene with lights, cameras, textures and many 3D models and displaying it on the screen takes as low as five lines of code. Advanced users can later accessing and modifying dynamically each element, by looking more deeply into the engine design or just use the highest-level instructions when they don't need unnecessary full control on specific details. One of the key points of MVisio making it very different from the game-oriented graphics products is the automatization of almost each feature through high-level methods taking care of everything but still offering experienced users to by-pass high-level calls to fine tune their needs. We can consider MVisio as a dual-head entity, exposing at the same time high and low level interfaces to the intrinsic features that can be accessed at the same time, according to the context and user needs. This is not only an advantage for new students who can have immediate results but also to experienced users to quickly build up a 3D scenario to use for their goals.

A MVisio-based application source-code for Windows is identical to its relative under Linux. Porting the same application from PC to mobile devices is just a matter of linking against different libraries and modifying a constant, switching from PC to CAVE just needs to specify the IP addresses of the CAVE-client computers. The following piece of C++ source code shows a very basic MVisio application supporting loading and rendering of a 3D scene on PDA, PC and CAVE:

```
//#define MV_PDA // <-- uncomment this for a PDA build
//#define MV_CAVE // <-- uncomment this for a CAVE build
#include <mvisio.h>

int main(int argc, int argv[])
{
    // CAVE build require to specify client PC IP addresses:
#ifdef MV_CAVE
    MVCLIENT *front = new MVCLIENT();
    front->setIP("192.168.0.1");
    front->setID(MV_FRONT);

   MVCLIENT *right = new MVCLIENT();
   right->setIP("192.168.0.2");
   right->setID(MV_RIGHT);

   // Etc...
#endif

    // Initialize the graphics engine:
    MVISIO::init();

    // Load full scene (textures, lights, models, etc.):
    MVNODE *scene = MVISIO::load("bar.mve");

    // If in the CAVE, update user head coordinates
    // for correct projection computation:
#ifdef MV_CAVE
    MVCLIENT::putUser(1.175f, 1.6f, 1.25f);
#endif

    // Display the scene:
    MVISIO::clear();
    MVISIO::begin3D();
       scene->pass();
    MVISIO::end3D();
    MVISIO::swap();

    // Free everything:
    MVISIO::free();
}
```

MVisio also natively satisfies most of the recent computer graphics standards, featuring direct support for complex model loading, advanced shading techniques and post-processing effects, skinning and animations, terrain rendering, integrated GUI systems, video2texture, etc. The user can just decide to activate or load one of these items and let MVisio automatically manage everything, or specifying each parameters through a very wide set of parametrization options featured in each class.

Another advantage of MVisio is the coherence of the design. Each element, either 2D (for the graphics user interface) or 3D, derives from the same base structure and exposes the same functionalities, thus reducing learning time for understanding how each MVisio object works and reducing code sizes. Advanced users can create and add new objects to MVisio by simply deriving from this base class, as shown in the following source code example. Users just need to implement the *render*() method that will be called by MVisio to display the object, when other functionalities like scene-graph compatibility and instancing will be natively managed by MVisio, reducing user tasks to perform when implementing new entities:

```
typedef class MY_SKYBOX : public MVNODE
{
public:
    MY_SKYBOX() { type = MV_NODE_CUSTOM; }

    bool render(void *data = 0)
    {
        MVELEMENT *element = (MVELEMENT *) data;

        // Scenegraph node base position:
        glLoadMatrixf((float *) element->getMatrix());

        // Setting OpenGL flags for this object:
        MVOPENGL::blendingOff();
        MVMATERIAL::reset();
        MVSHADER::disable();
        MVLIGHT::lightingOff();
        // Etc...

        // Perform native OpenGL calls:
        glColor4f(1.0, 1.0, 1.0,1.0f);
        glTranslatef(position.x,position.y,position.z);
        glScalef(0.5f, 0.5f, 0.5f);

        // Display the skybox with OGL triangles
        glBegin(GL_TRIANGLES);
        glVertex3f(20.0f, 20.0f, 20.0f);
        // Etc...

        return true;
    }
} MY_SKYBOX;
```

3.2 Pedagogical and Corollary Tools

The MVisio graphics engine is the core product of our framework, but not the only one. When MVisio is aimed mainly to concrete practice and developing of applications, we also created a set of corollary tools oriented to aid a more *ex cathedra* teaching approach. The Mental Vision platform is then completed by three additional tools: a set of pedagogical modules, a series of tutorials, and other utilities to reduce and simplify development times with MVisio.

Pedagogical modules are compact and interactive demonstrators created to practically illustrate a specific algorithm, method or technique introduced during the class. Modules are concretely executable files created on the top of the MVisio engine (thus inheriting the same robustness and smoothness) working as small stand-alone applications that both students and teachers can download and use as learning support. Moreover, modules can be used on handheld devices that assistants may use during practical sessions to directly illustrate additional explains on the desk. Modules are distributed along with their source code that can be used by learners as an example of uses of the MVisio graphics engine itself.

Tutorials refer to the use of the graphics engine. They are a suite of HTML pages with downloadable examples (conceptually similar to the very popular tutorial sites like Nehe[11] or Gametutorials[12]) illustrating step by step how to start working with MVisio. Tutorials are used for introductory notions and first practice with MVisio, while modules aim to cover more specific aspects by letting users reading their source code to find out how an algorithm or technique has been implemented.

Besides the graphics engine, modules and tutorials, the Mental Vision framework comes with a series of extra tools like Autodesk[13] 3D Studio Max plugins or FBX converters to easily import 3D contents from third-part products into MVisio. Extra tools include also optional classes for using a joypad to control 3D cameras or some basic artworks to improve the aspect of the embedded GUI.

4 Implementation

This section gives an architectural and technical overview about the main components of the Mental Vision platform, namely the MVisio 3D graphics engine and pedagogical demonstrators.

4.1 System Architecture

The Mental Vision framework is a multi-platform and multi-device system featuring a 3D graphics engine with an unique and same interface independently from the operating system or device we want to use, and cross-platform and

[11] `http://nehe.gamedev.net`
[12] `http://www.gametutorials.com`
[13] `http://www.autodesk.com`

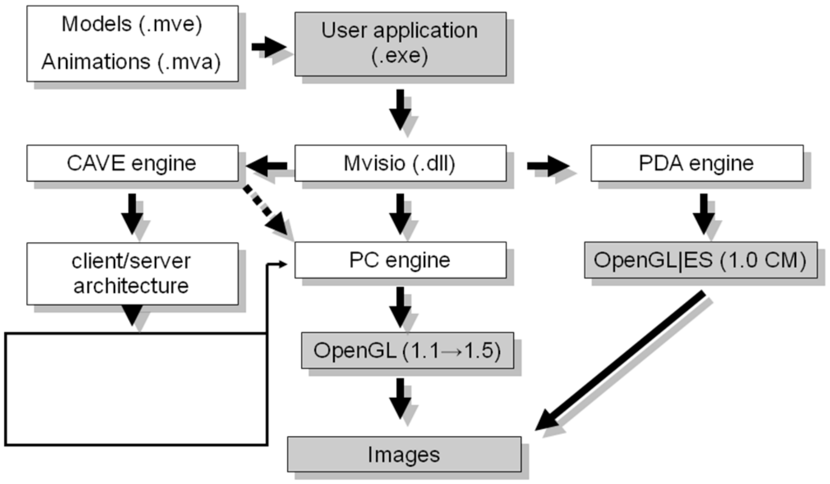


Fig. 2. MVisio multi-device rendering pipeline overview

cross-device pedagogical demonstrators and tools. Users adopting MVisio as graphics engine just need to link their code with the appropriate library created for each OS/device and let our software manage the rest. Students practicing or reviewing class notes through our modules just need to pick the appropriate version supporting their platform and run it.

MVisio automatically tunes its internal pipeline through different code-paths, according to the context. For example a typical PC application (x86 architecture on a Linux or Windows based system) directly uses the graphics hardware available to perform the best rendering approach. CAVE systems require a more complex approach: a local instance of MVisio (that runs on the same machine the user is developing on) becomes the server of the CAVE system. This server version of MVisio communicates with the several CAVE client machines, each one running a daemon service waiting requests from a server. Locally, each CAVE client (one per wall) starts a MVisio version for PC and reproduces all the high-level methods invoked server-side by the user. We can consider MVisio for CAVE as a remote playback of methods called on the server PC (see figure 2).

MVisio for mobile devices is very similar to the MVisio for PC version, but is optimized for fixed-maths ARM processors and uses only the basic graphics functionalities of the engine, that is the only ones that may run on the very limited resources available on handheld devices. It is important to mention that the only differences for the end user between running his/her application on a PC, CAVE or mobile device concern only the version of MVisio to link with: no other modifications are required.

4.2 Technical Details

The Mental Vision software is entirely written in C++. The graphics engine is distributed as a stand-alone dynamic-link library (DLL) to link against using its

.h and .lib files. The entire API is class-oriented and uses an internal resource manager releasing the users from the need of freeing allocated entities.

MVisio uses a slightly customized SDL version (Simple DirectMedia Library[14]) for basic platform independent output window creation, event and threading management. Low-level graphics rendering is performed via OpenGL[15] on personal computers/CAVEs and through OpenGL|ES[16] on mobile devices.

On PCs and CAVEs, MVisio supports OpenGL version 1.1 up to 1.5, by automatically compiling and using different code-paths according to the hardware quality the engine is running on. For example on a full OpenGL 1.5 compliant PC hardware skinning, soft-shadowing, per-pixel lightning and different post-processing effects (like depth of field and bloom-lightning) are activated.

On mobile devices (based on Windows CE 4 or better) MVisio comes with two different versions: a generic one, using a software implementation of OpenGL|ES (made by Hybrid) and a hardware accelerated version, running on PowerVR MBX-lite[17] graphics boards. In both cases we used OpenGL|ES 1.0 Common Lite profiles.

MVisio for CAVEs uses TCP/IP communications for data transmission between server and clients (refer to [4] for more details).

5 Cases of Study

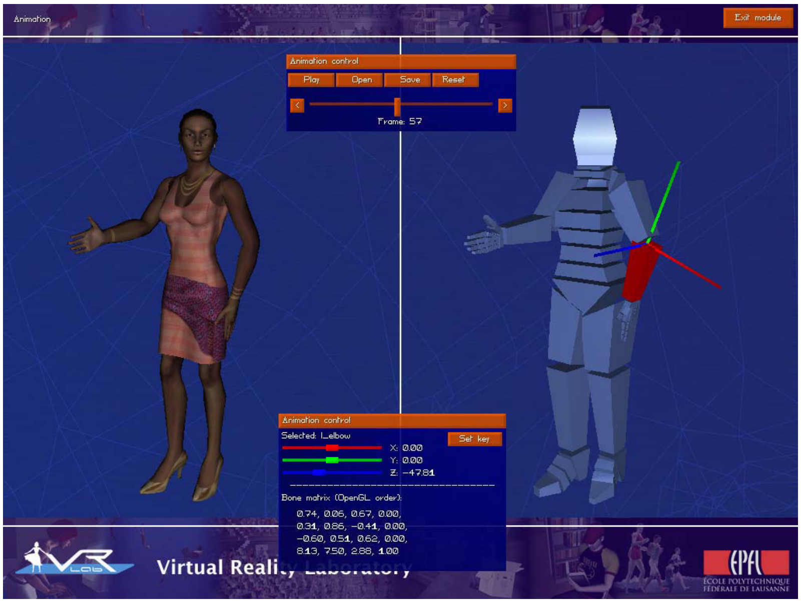
This section presents several cases of study related to concrete uses of our platform, ranging from educational/practicing to scientific applications. Each case will be briefly summarized and followed by a discussion about the advantages brought by the adoption of the Mental Vision framework.

5.1 Mental Vision for Education During the Lessons

Teaching topics like splines, clipping planes or vertex skinning may be a difficult task because of the abstract notions required by the learning process. Teachers often recur to schematics or videos to support their explains with visual feedback. Despite the clearness contribution yielded by images to the learning process, practice and interactivity are not covered by these supports.

Pedagogic Modules. To improve these aspects of teaching, we developed a series of interactive modules to be used during lessons to concretely and dynamically show how an algorithm, technique or concept work. Teachers have a more robust and dynamic support to visualize on a large screen notions they're explaining, other than a blackboard or a video. Modules allow both teachers and students to directly interact with the topic discussed and have a direct relationship between the modifications they apply and the results they get (What

[14] `http://www.libsdl.org`
[15] `http://www.opengl.org`
[16] `http://www.khronos.org/opengles`
[17] `http://www.imgtec.com`



```
Selected: l_elbow
X: 0.00
Y: 0.00
Z: -47.81
-----------------------------------
Bone matrix (OpenGL order):
    0.74, 0.06, 0.67, 0.00,
    0.31, 0.86, -0.41, 0.00,
    -0.60, 0.51, 0.62, 0.00,
    8.13, 7.50, 2.88, 1.00
```

Fig. 3. Animation module: students can create, manipulate and export short animations for later usage within the MVisio 3D engine

You See Is What You Get approach). Moreover, modules can be downloaded and used at home for reviewing the class content and their source-code read for potential implementations during practical work sessions: being both modules and practical sessions based on the top of the MVisio engine, the course get a coherent guiding thread between the theoretical aspects and the practical ones, reducing the patchwork-like approach many courses suffer from using some tools during the lessons, other ones in the class-notes and different ones again during workshops and practice. Modules show also a great utility in e-learning contexts, offering students remotely following the class to still be able to practice and repeat the experiences taught.

For example the module about skinned animation allows students to create short animations by setting different key-postures on a time-table affecting a virtual character (see figure 3). This way, teachers don't need to rely on cumbersome and expensive software like 3D Studio Max or Maya to demonstrate and practice with this topic. Furthermore, animations created through this compact module can be saved and exported on a file for later usage in MVisio for practical work or class projects, connecting theoretical course lessons with practical sessions.

Other modules also feature this editing options, like the particle engine and terrain engine related ones: these modules can be used as teaching support during the lesson time and as particle or terrain editors during the practical sessions. Users can export to a file the particle system or terrain created through the

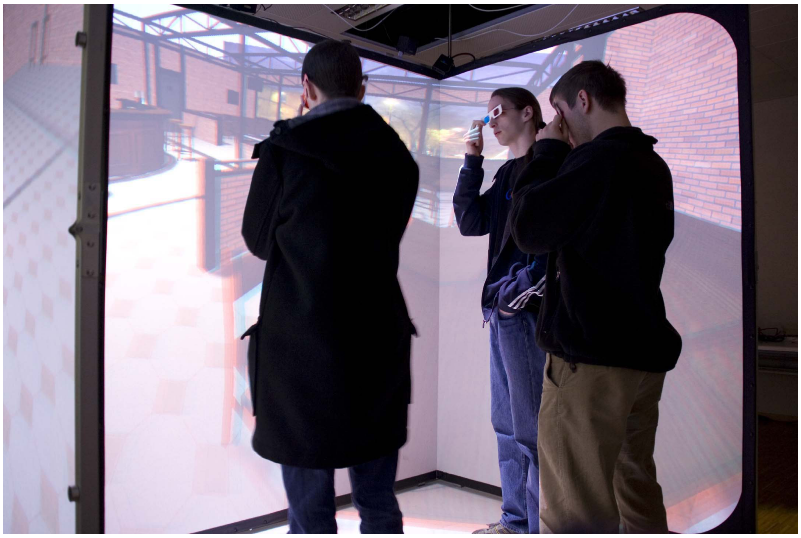


Fig. 4. Our CAVE used during young student visits to disseminate scientific technologies

module and load them directly into MVisio, thus reducing dependencies from external software and improving coherence with the theoretic aspects.

Modules are even more connected between *ex cathedra* lessons and practical sessions by their mobile versions, running on handheld devices. During practical sessions, assistants can freely sit at the student desks with their handheld devices and support their additional explains by reusing the mobile version of a module as a portative blackboard.

CAVE and Dissemination. We integrated a visit to our laboratory to the course plan in order to let students seeing and experiencing with our VR equipments. Particular attention has been used for the CAVE part of this visit, by using it to furthermore fix concepts like spatial level of detail and terrain rendering techniques. Students can enter the CAVE and experience these techniques by seeing them concretely used around them, with stereographic and immersive rendering. CAVE systems have the advantage of being very curious and rare devices that will attract students attention and give them a long term remember of the experience.

We also use our CAVE as attraction during public demonstrations for external visitors, like secondary or high school students, for disseminating technologies we use in our field through user-friendly and interesting demos (see figure 4).

Thanks to the automatic multi-device support of MVisio and the easiness of its calibration support for the CAVE rendering, it is straightforward and very time effective to run, maintain or improve such demos: in less than 5 minutes the entire system can be started, calibrated and ready for visitors.

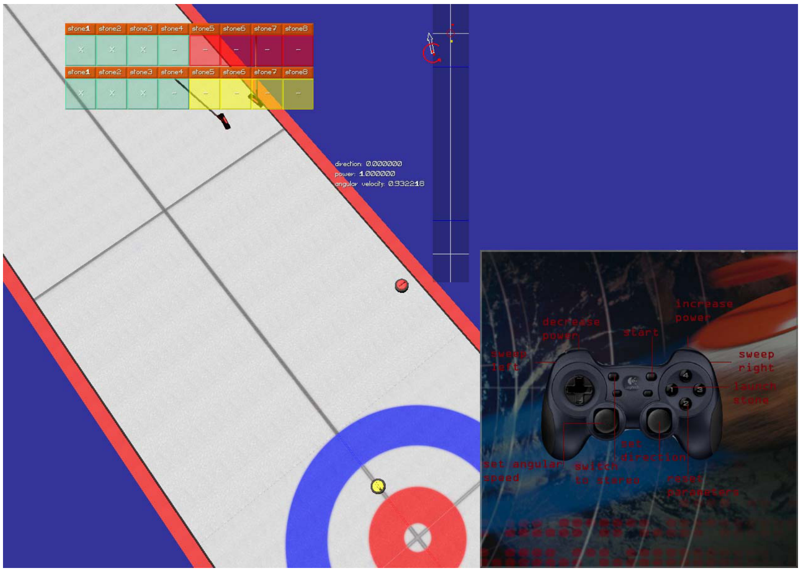


Fig. 5. A curling simulation game made by students with animations, dynamic lightning and GUI

5.2 Mental Vision for Practical Works

Theoretical classes are coupled with practical sessions where students may concretely apply the knowledge introduced during the different lessons. We already observed in [12] that proposing the creation of little game-like applications, including most of the VR and CG related aspects taught during the class, was an interesting motivating factor. We improved this aspect by introducing the MVisio 3D graphics engine as the standard tool for semester projects and workshops.

The Curling. Students of a virtual reality course were asked for developing a simulation of a curling match (see figure 5). With such a project, we aimed to make students practice with several aspects of virtual reality by integrating in the same work realtime graphics, basic physics, stereographic rendering, force feedback (through a vibrating joypad) and some game mechanics to implement curling rules. The practical sessions lasted one semester in reason of one hour per week and projects were done by groups of two students. We gave for the first time MVisio to students, a basic DirectInput library, 3D models and some guidelines for the physics and game related aspects.

At the end of the semester we evaluated about twenty projects. Compared to previously years projects, made by using other tools or letting students to freely choose which software adopt, a clear improvement in both the quality and completeness of the work were noticeable. Despite the low amount of time accorded weekly on this task, they managed to take care of all the aspects and improved their global vision about the creation of a VE. The immediate results available

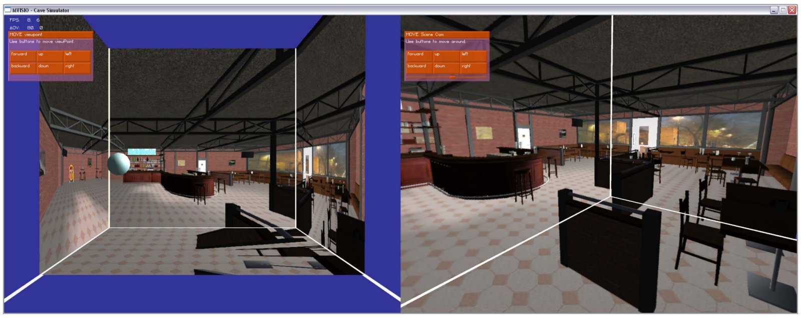


Fig. 6. Virtualized CAVE to locally preview on a single PC an immersive application developed with MVisio

through the lightweight MVisio interface also kept their motivation high, reducing the gap from the documentation reading to the first results considerably: for example at the end of the first session students managed to display and move the different scene elements on the screen. Thanks to the MVisio API compactness, only few lines of code are required to perform such tasks, thus reducing considerably the amount of information users have to understand in order to be operative.

Semester Projects. We have been offering bachelor/master level projects about specific VR/CG topics for students who decide to pass their semester projects in our laboratory. Such project topics may be very heterogeneous, like implementing state-of-the-art techniques in our software or helping assistants in their researches. For example we offered many projects related to mobile devices or our CAVE, like a 3D GPS-like handheld remote controller to perform teleoperations on a real blimp or implementing a full immersive game in the CAVE. Unfortunately, we just have a single CAVE and very few mobile devices that may be shared among all the persons needing them. Thanks to the multi-device portability of MVisio, we can get rid of this problem by letting users work on the PC version and just run their projects later on the hardware they need. For CAVE users requiring a wider overview we also developed a Virtual CAVE (see figure 6) featuring a single PC preview about how their application will look like once ran on the real device.

5.3 Mental Vision for Research and Prove of Concept

We exposed so far minor uses of the MVisio engine on semester projects, course practical works or similar: in the following examples we describe deeper and more long-term applications of MVisio for thesis research projects.

MHaptic. MVisio has been integrated into a haptic engine (called MHaptic [18]) to manage the visual output of the system, featuring the Immersion[18] Haptic Workstation and a user-worn HMD (see figure 7).

[18] `http://www.immersion.com`

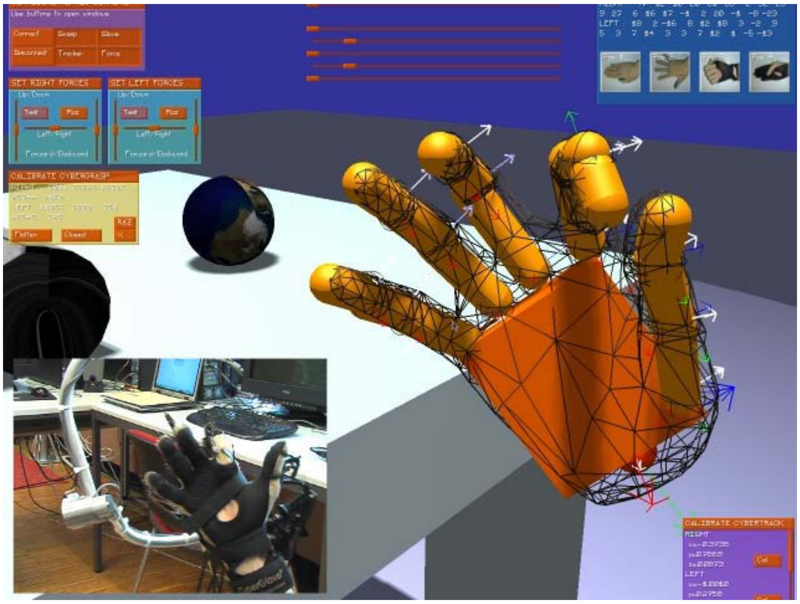


Fig. 7. MHaptic haptic engine using MVisio for visual rendering

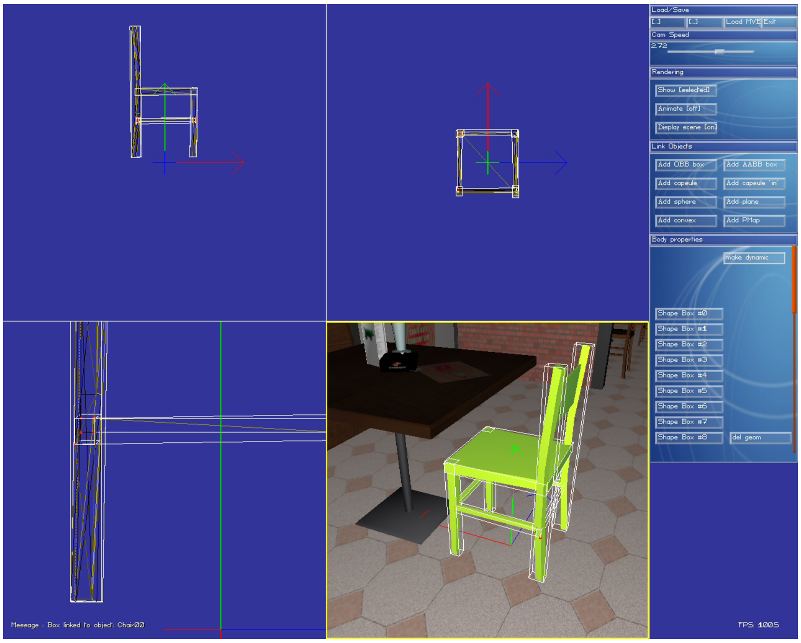
MHaptic relies on an external editor, MHaptic Scene Creator, which is a 3D Studio Max-like software to add haptic proprieties to a 3D scene (like weight, collision detection, physics, etc.). This software has been developed by largely using the GUI system and advanced functionalities of MVisio (see figure 8).

The MHaptic project shows how far MVisio can be used not only to simplify teachers and students needs but also to develop very complex and advanced applications.

Human Animation. MVisio native support for large screens, CAVEs and head-mounted displays has been widely used in [19] to test and validate inverse kinematics reaching techniques authors introduced. Thanks to the MVisio portability, they could repeat the experience by giving visual feedback to users on PC, through an HMD or in front of large screens with no need of major modifications to the applications they developed (see figure 9).

6 Conclusions and Future Work

This publication describes motivations, approaches and results we obtained by creating our graphics framework called Mental Vision. We used it for teaching, practice and research in the field of computer graphics and virtual reality. We cite many different cases of study where the MVisio graphics engine and pedagogical tools have been successfully used, showing advantages and benefits. The MVisio 3D graphics engine simplifies the creation of virtual environments



Message : Box linked to object: Chair00

FPS: 100.5

Fig. 8. MHaptic editor for adding haptic properties to virtual objects, entirely developed using MVisio and its GUI support

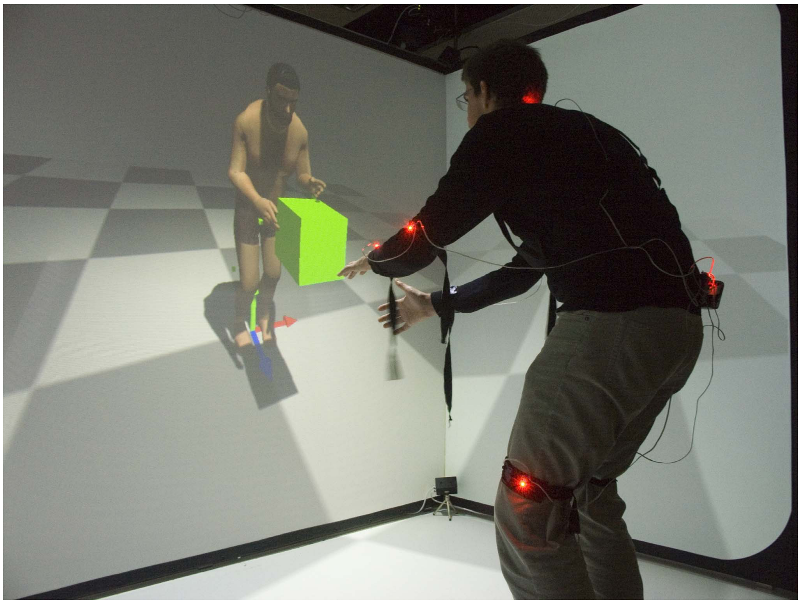


Fig. 9. User testing the different reaching techniques in the CAVE

and the adoption of less programmer-friendly platforms like handheld devices or CAVE systems. MVisio satisfies at the same time a very wide range of needs and perfectly fits into a series of different cases that usually would require more than a single software and approaches. Thanks to MVisio, our entire laboratory is working on the top of the same system, from researchers to students, improving inter-personal cooperations, reducing learning times and facilitating the maintaining of software written by other persons or members who left the crew.

We are now planning to make the CAVE and mobile device support more generic. CAVE support should become useable on CAVE installations other than ours, with an arbitrary shape and number of walls, while the hardware accelerated support for handheld devices should be extended to more recent mobile phones and personal digital assistants, supporting OpenGL|ES 1.1 or higher.

MVisio version 1.6.4 and pedagogical modules are freely downloadable for teaching and researching purposes on our website[19].

Acknowledgments. Mental Vision has been sponsored by the EPFL Funding Program for Teaching and Learning (FIFO). MVisio has been recently used and extended in the framework of the INTERMEDIA Network of Excellence.

References

1. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., Hart, J.C.: The cave: audio visual experience automatic virtual environment. Commun. ACM 35(6), 64–72 (1992)
2. Peternier, A., Thalmann, D., Vexo, F.: Mental vision: a computer graphics teaching platform. In: Pan, Z., Aylett, R.S., Diener, H., Jin, X., Göbel, S., Li, L. (eds.) Edutainment 2006. LNCS, vol. 3942, pp. 223–232. Springer, Heidelberg (2006)
3. Peternier, A., Vexo, F., Thalmann, D.: Wearable Mixed Reality System In Less Than 1 Pound. In: Proceedings of the 12th Eurographics Symposium on Virtual Environment (2006)
4. Peternier, A., Cardin, S., Vexo, F., Thalmann, D.: Practical Design and Implementation of a CAVE Environment. In: International Conference on Computer Graphics, Theory and Applications GRAPP, pp. 129–136 (2007)
5. Kot, B., Wuensche, B., Grundy, J., Hosking, J.: Information visualisation utilising 3d computer game engines case study: a source code comprehension tool. In: CHINZ 2005: Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction, pp. 53–60. ACM, New York (2005)
6. Soh, J.O.B., Tan, B.C.Y.: Mobile gaming. Commun. ACM 51(3), 35–39 (2008)
7. Towle, T., DeFanti, T.: Gain: An interactive program for teaching interactive computer graphics programming. In: SIGGRAPH 1978: Proceedings of the 5th annual conference on Computer graphics and interactive techniques, pp. 54–59. ACM, New York (1978)
8. Song, W.C., Ou, S.C., Shiau, S.R.: Integrated computer graphics learning system in virtual environment: case study of bezier, b-spline and nurbs algorithms. In: Information Visualization, 2000. Proceedings. IEEE International Conference, pp. 33–38 (2000)

[19] `http://vrlab.epfl.ch/~apeternier`

9. Meeker, P.H.: Introducing 3d modeling and animation into the course curriculum. J. Comput. Small Coll. 19(3), 199–206 (2004)
10. Hill, J.M.D., Ray, C.K., Blair, J.R.S., Curtis, A., Carver, J.: Puzzles and games: addressing different learning styles in teaching operating systems concepts. SIGCSE Bull 35(1), 182–186 (2003)
11. Becker, K.: Teaching with games: the minesweeper and asteroids experience. J. Comput. Small Coll. 17(2), 23–33 (2001)
12. Gutierrez, M., Thalmann, D., Vexo, F.: Creating cyberworlds: experiences in computer science education. In: International Conference on Cyberworlds, 2004, pp. 401–408. Virtual Reality Lab., Swiss Fed. Inst. of Technol, Lausanne, Switzerland (2004)
13. Korte, L., Anderson, S., Pain, H., Good, J.: Learning by game-building: a novel approach to theoretical computer science education. In: ITiCSE 2007: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, pp. 53–57. ACM, New York (2007)
14. Clevenger, J., Chaddock, R., Bendig, R.: Tugs: a tool for teaching computer graphics. SIGGRAPH Comput. Graph. 25(3), 158–164 (1991)
15. Coleman, R., Roebke, S., Grayson, L.: Gedi: a game engine for teaching videogame design and programming. J. Comput. Small Coll. 21(2), 72–82 (2005)
16. Tori, R., Jo ao Luiz Bernardes, J., Nakamura, R.: Teaching introductory computer graphics using java 3d, games and customized software: a brazilian experience. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Educators program, p. 12. ACM, New York (2006)
17. Wilkens, L.: A multi-api course in computer graphics. In: CCSC 2001: Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges, USA, Consortium for Computing Sciences in Colleges (2001)
18. Ott, R., De Perrot, V., Thalmann, D., Vexo, F.: MHAPTIC: a Haptic Manipulation Library for Generic Virtual Environments. In: Haptex 2007(2007)
19. Peinado, M., Meziat, D., Maupu, D., Raunhardt, D., Thalmann, D., Boulic, R.: Accurate on-line avatar control with collision anticipation. In: VRST 2007: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, pp. 89–97. ACM, New York (2007)