GaDeVi
Game Development Integrating Tracking and Visualization Devices into Virtools

Ricardo Aguiar[1], João Madeiras Pereira[2]
[1] *IST/UTL, Lisbon, Portugal*

[2] *IST/UTL INESC-ID, Lisbon, Portugal*

José Braz
EST/IPS, Setúbal, Portugal

Keywords: Virtual Reality, Immersive Devices, Game Engine, Computer Game, Interactivity, Immersion.

Abstract: Computer and console games are increasingly using non-computer conventional input/output devices. This work's main goal was to create an interactive 3D application with Virtools, a development tool that appeals to the player dexterity and astuteness through the use of several immersive, non-computer conventional devices like magnetic trackers, data gloves and VR glasses. The present paper describes the integration of interaction and visualization devices into Virtools as well as the development of two games using the VR environment. To finalize the assessment of the performance measurements and user tests is given.

1 INTRODUCTION

Nowadays, computer and console games are an increasingly profitable market (ELSPA, 2003). In fact, one of the main driving factors for hardware and technological evolution in IT occur because of the increasing need of greater capacity for processing data to simulate high complexity physical effects and/or offer a realistic visual quality in a game. Videogame programming is also a very interesting area of development. Trying to give the gamer a complete immersive experience becomes a challenge (Goebel et al, 2001). However most of today's games are limited to standard input/output devices, like a mouse and a keyboard on a PC or a game pad on a game console. This last statement was true in 2004 (Hinckley et al, 2004) and still true today, even if it should be mentioned that the Wii platform is the exception in the console market. Moreover the game 3D content is almost always displayed into a 2D screen such as CRT or LCD.

The goal for this work is to create games with professional developing tools that offer new ways of interaction with the player through the use of immersive devices. The visualization will not be made by a normal 2D screen but through a set of VR Glasses in order to give a more realistic awareness of all three dimensions.

Virtools 4.0 was the main platform to develop the games along with Visual Studio.Net (Microsoft) which was used to create new features/plug-ins. The following immersive devices where programmed into the games: Magnetic tracker Fastrak (Polhemus), Data Gloves Ultra (5DT) and Z800 3D Visors (eMagin). Blender and 3D Studio Max (Autodesk) were used as modelling and skinning tools.

2 TECHNOLOGY DESCRIPTION

This section introduces all three devices listed above and provides a brief description of Virtools.

2.1 Polhemus Fastrak

A tracker's purpose is to provide real time information of the location and movement of a receiver module relatively to an emitter module. Relative spatial coordinates (x, y, z) and three Euler

Aguiar R., Madeiras Pereira J. and Braz J. (2009).
GaDeVi - Game Development Integrating Tracking and Visualization Devices into Virtools .
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications*, pages 313-321
DOI: 10.5220/0001755003130321

angles (azimuth, elevation, roll), thus six degrees of freedom, are given to the user. A tracker device has three main components (Polhemus, 2005), an interface module, also called filter, an emitter module and finally a receiver module.

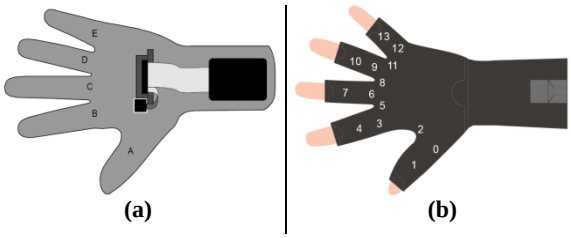

Figure 1: 5DT Data Gloves. (a) 5 Sensor Model, (b) 14 Sensor Model.

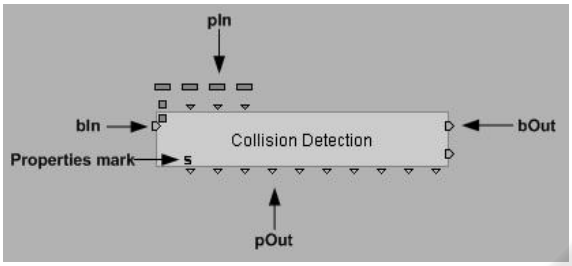

Figure 2: Behavioural Building Block Layout.

2.2 5DT Data Gloves

Data gloves capture hands motion. Several sensors are placed into specific positions of the glove in order to capture finger motion, process this data and supply it to the computer. The models which are used in this work are *Left Handed 5 5DT Data Glove Ultra* and *Right Handed 14 5DT Data Glove 14 Ultra* (5DT, 2004). These models main differences are in the number of sensors available at the glove.

2.3 eMagin Z800 3D Visor

A pair of VR Glasses is used instead of a 2d screen enhancing depth perception which is achieved using stereoscopy.

Stereoscopy is defined as the creation of a three-dimensional image by presenting two slightly different images to each of the human eye. Several algorithms exist that perform stereoscopy. The most popular techniques are anaglyph, passive stereo and active stereo. With anaglyph, an image is constructed with two overlapped layers, each one filtered with a different colour. The viewer must wear a special set of glasses in which each lens has a different colour.

Passive stereo principle uses light polarization to transmit a rendered stereo pair. The image is recombined into the glasses lens through filters. Active stereo, implemented in 3D Visors, transmits two different images, with the Graphics Card Vertical Sync controlling its frequency. Each one of those images is sent to micro displays located in left/right eye.

3D Visor also provides tracking information using accelerometers and an internal gyroscope, making possible to know the headset orientation and type of movement.

2.4 Dassault Systèmes Virtools

Virtools, a game engine, was the main platform used in this game development. It has the advantage of creating interactive 3D content for computers and consoles in a fast and intuitive way using Product Context Scenario "PCS" (Mc Carthy et al, 2006), a Virtools proprietary programming paradigm. PCS allows a programmer to create applications using only drag & drop operations of special entities called behavioral Building Blocks (BB). A BB has the aspect shown in Figure 2.

It has five main components:

- Behavioural Inputs (bIn)
- Behavioural Outputs (bOut)
- Parameter Inputs (pIn)
- Parameter Outputs (pOut)
- Properties marks.

The bIn is where the BB receives an activating signal to start its internal processing; bOut generates an activation signal when the BB has finished which is then propagated to other BB. Through pIn and pOut, data is transmitted between BB, receiving external data in pIn and transmitting data through pOut. The "Properties mark" indicates if:

- BB may send/receive messages through scripts
- Parameters processing may be changed
- New bIn, bOut, pIn, pOut may be inserted into the BB.

The Virtools layout has three main components, 3D Layout, Building Blocks & Data Resources, and Manager & Schematic Window, as shown on Figure 3.

In 3D Layout, common abstract objects in a typical game engine like cameras or lights, may be created and placed in a game. 2D and 3D content of a Level may be manipulated in directly at the 3D Layout, using transformation operations. It is also

possible to change camera parameters like field of view, panning, etc.

Inside *Building Blocks & Data Resources*, all available behavioural Building Blocks are listed and are ready to use, as well as a set of native Virtools resources like images, sounds and so on as shown on Figure 4.

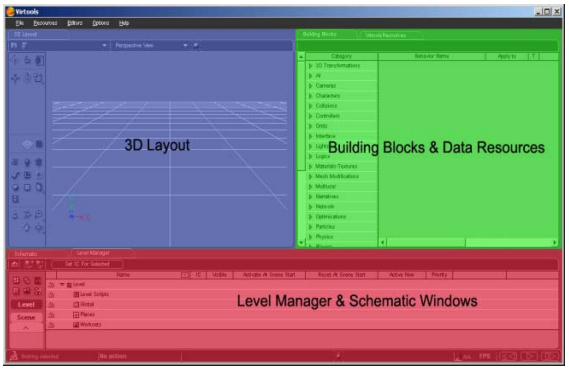

Figure 3: Virtools Main Layout.

Finally in Level Manager & Schematic Window, all the Level objects are listed in the Level Manager section, allowing the programmer to access their setup. It is also possible to create new scripts and logical entities like Arrays, Groups, etc. In the Schematic Window scripts are programmed by filling them with BBs, Parameter Operations and Variables as shown in Figure 5.

3 DEVICES INTEGRATION

Virtools standard version does not support the set of immersive devices listed above, so it is necessary to integrate them. This process is made using the Virtools SDK and the devices specific Application Programming Interfaces. New BB and Managers are created with SDK using C++. To develop a new BB it is necessary to establish the number of bIn/ bOut, the number and type of pIn/pOut, types of properties marks and define an execution function which has the desired BB internal processing and bOut activation. A Manager, manages data resources, external communications, etc. The order through which the managing process is performed must be defined in a Manager. It may be done after or before a frame is processed.

3.1 Polhemus Fastrak Integration

Polhemus Fastrak API programming interface, FTAPI, allows the connection and communication between the PC and tracker. After an established connection, the programmer may request data such as relative position in rectangular coordinates [x, y, z] and orientation in Euler angles [Azimuth, Elevation, Roll] of one of four possibly connected receivers to a unique transmitter.

A single behavioural Building Block is developed with FTAPI, which is responsible by managing the trackers communication with the PC and supply its data to the programmer. However, this approach decreases drastically the applications overall performance with frame rates falling down from 60 to 5-10 fps. To overcome this lack of performance, a Manager is created and made responsible for the communication with the tracker and supplying the requested data to a newly developed BB, named TrackerBB The programmer may change from which receiver he wants to obtain tracking data inserting a valid receiver id into StationId pIn as shown in Figure 6.

The Manager, before each frame is processed, requests new data from the specified receiver and make it available to TrackerBB to use it in a composition. The tracking data from the BB may be used through its pOuts which specify, from left to right:

Vector with relative rectangular receiver coordinates [x y, z]

- Receiver's relative Azimuth angle orientation
- Receiver's relative Elevation angle orientation
- Receiver's relative Roll angle orientation
- Connection Status
- Tracker Debug Information
- Tracker internal latency time, first 32 bits
- Tracker internal latency time, last 32 bits

Using the BB - Manager Scheme increases the frame rate up to 30 fps.

3.2 5DT Integration

A similar plug-in is developed for the data gloves. A Manager is created with Virtools SDK and *FGlove* API. As in the tracker, it manages the communication and data flow from the gloves and a newly developed BB named GloveBlock.

GloveBlock has two pIn: Mode and gloveId. Mode indicates if the sensors outputs are raw or auto-calibrated values. If raw data is selected the values are given in a range of 0-4095 (12 bit resolution) in which 0 is no sensor flexure and 4095 is total sensor flexure. In case calibrated values

mode is selected the values range is 0-1, with 0 indicating no sensor flexure and 1 indicating total sensor flexure.

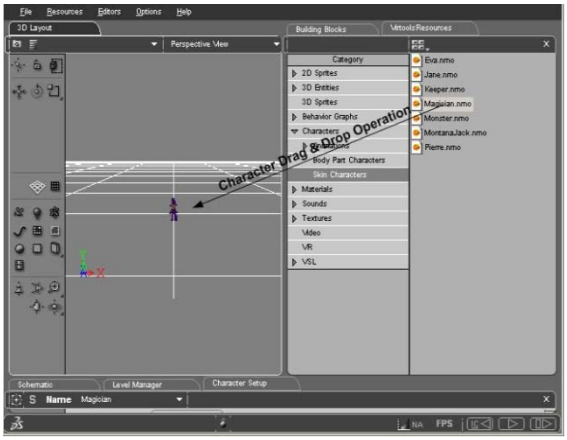

Figure 4: Resources Drag & Drop.

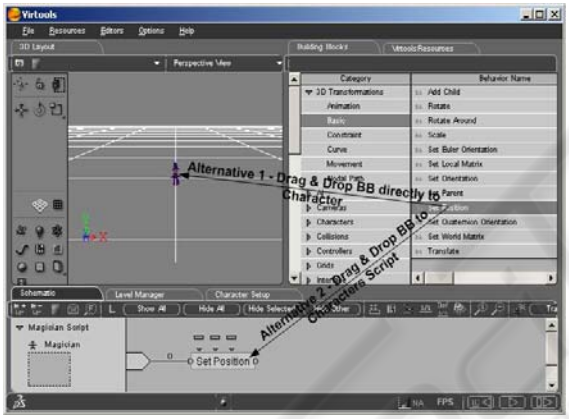

Figure 5: BB Drag & Drop.

Figure 6: Tracker behavioural Building Block.

Figure 7: Glove behavioural Building Block.

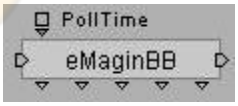

Figure 8: 3D Visor behavioural Building Block.

The data that can be retrieved by pOuts is, from left to right:

- Sensor 0 flexure or Sensor A flexure (fig 1)
- Sensor 1 flexure or Sensor A flexure (fig 1)
- Sensor 2 flexure (fig 1)
- Sensor 3 flexure or Sensor B flexure (fig 1)
- Sensor 4 flexure or Sensor B flexure (fig 1)
- Sensor 5 flexure (fig 1)
- Sensor 6 flexure or Sensor C flexure (fig 1)e
- Sensor 7 flexure or Sensor C flexure (fig 1)
- Sensor 8 flexure (fig 1)
- Sensor 9 flexure or Sensor D flexure (fig 1)
- Sensor 10 flexure or Sensor D flexure (fig 1)
- Sensor 11 flexure (fig 1)
- Sensor 12 flexure or Sensor E flexure (fig 1)
- Sensor 13 flexure or Sensor E flexure (fig 1)
- Hand gesture recognition
- Handedness of the glove
- Glove Model Description
- Number of gloves connected
- Type of gloves connected
- Internal latency time, first 32bits
- Internal latency time, last 32bits

3.3 eMagin Z800 Integration

3D Visors has two components to be integrated with Virtools. The first one is headset orientation given from its internal gyroscopes and the second one is stereoscopic visualization to enhance depth perception.

Headset Orientation - Like in the gloves and magnetic tracker, a Manager and a behavioural Building Block, eMaginBB**,** are created using Virtools SDK and eMagin SDK. The Manager handles communication with 3D Visors and retrieves its orientation data to make it available to eMaginBB.

There is only one pIn: PollTime. It allows the programmer to change the time interval in which a callback function is internally called in the manager to collect new orientation data from the 3D Visor device (Figure 8)

The pOuts, from left to right, have the following data:

- Roll angle
- Pitch angle
- Yaw angle
- Internal latency time, first 32bits
- Internal latency time, last 32bits

Visualization - Virtools already have a plug-in, VR-Pack (5DT, 2004), that deals with stereoscopy. However, a set of external files must be configured in order to activate stereoscopy into an application. Also the game camera must be configured in a way that its view frustum becomes asymmetrical and provide a correct perception of stereoscopy.

Convergence Plane: Plane where objects left and right images converge

Parallax: Difference in centimetres between scenes projected in left and right images

IPD: Interpuppilar distance

Using asymmetrical view frustums cameras makes the created depth perception effect the most immersive possible. These special cameras are known as Projection Referential Cameras and, instead of using a position, orientation, aspect ratio, field of view, near and far clips, use a position, 3D Sprite (eMagin, 2005) and near and far clips. A 3D Sprite is a basically a 2D image/quad inside Virtools with a certain position and orientation.

If conventional cameras were used into with stereoscopy the result would be as shown in Figure 12.

Notice that the convergence plane is located at the infinite and the parallax is always negative, meaning that all the objects are perceived before the convergence plane, which is obviously wrong. A well balanced stereoscopic image must contain objects with a negative and positive parallax, so that they are perceived after and before the convergence plane. Also some of the objects may have a zero parallax which indicates that they intercept the convergence plane.

4 GAME DEVELOPMENT

A brief description will be made about the games that were developed during this work. A set of core scripts were created to acquire the data from all the immersive devices and to configure active stereoscopy. Both the data from the tracker, glasses gyroscope and the gloves passes through a simple software low-pass filter, which minimizes the data's high-frequency noise. It calculates an average value from a pre-defined number of samples taken from the BB's TrackerBB, eMaginBB and GloveBlock.

Position and orientation data is retrieved from the tracker manager into trackerBB (Figure 13). This data is passed through the filter and a new position and orientation are calculated and applied to the desired 3D object to control.

For the data gloves, flexure data passes through a low pass-filter similar to the one described above, a flexure variation is then calculated for each sensor and applied to the virtual hands (Figure 14). If activated, the player camera position will be translated accordingly to the value from the gesture pOut.

The pseudo-code for 3D Visor (Figure 15) is similar to the magnetic tracker, but it only calculates an orientation with yaw and pitch angles since roll angle rotation is not usually used in a camera orientation.

For the 3D Visor, besides the external configuration files, the BBs VR Get Config Token, VR Stereo Settings and VR Set Projection Ref are used to define the stereoscopy with a custom created Projection Camera (Nahon et al, 2006).

Two games were created using these core scripts, PongVR, a table tennis simulator and PianoVR, a virtual piano.

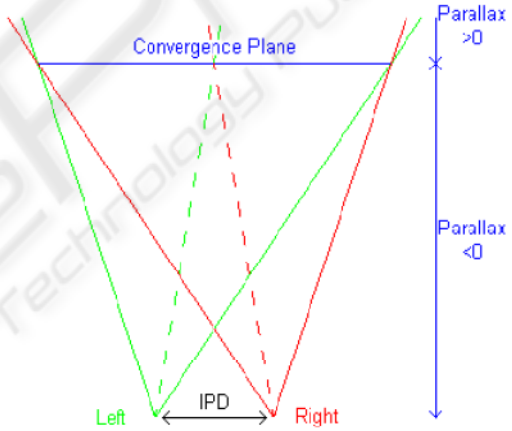

Figure 9: Correct Stereoscopy.

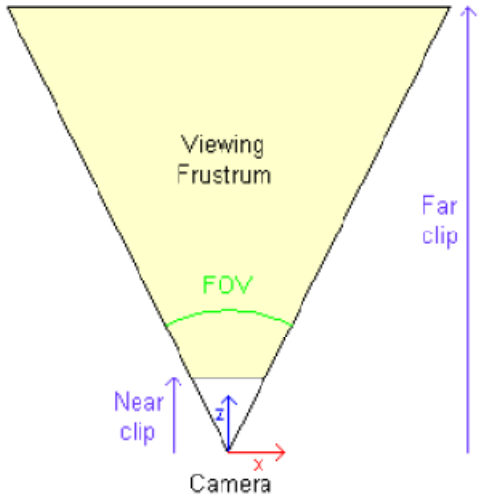

Figure 10: Standard Camera.

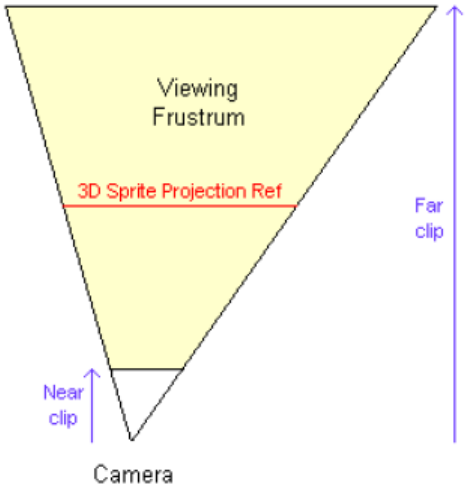

Figure 11: Projection Referential Camera.

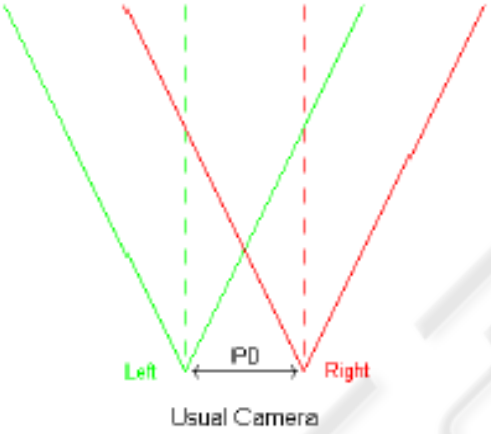

Figure 12: Stereoscopy with standard cameras.

```
1. Get Position and orientation data
      from magnetic tracker
2. Calculate TrackerBB execution time
3. Attenuate high-frequency noise
      with low-pass filter
4. Data insertion on a vector with a
      pre-defined size
5. Eliminate oldest value if vector
      full
6. Calculate an arithmetic average
      value from vector data
7. Calculate new orientation
8. Apply new orientation to
      hand/racket using local
      coordinate system
9. Calculate new position
10.Apply new position to hand/racket
      using world coordinate system
```

Figure 13: Magnetic tracker pseudo code.

```
1. Obtain gloves flexure data/gloves
      gesture
2. Calculate GloveBlock execution
      time
3. Attenuate high-frequency noise
      with low-pass filter
4. Data insertion on a vector with a
      pre-defined size
5. Eliminate oldest value if vector
      full
6. Calculate an arithmetic average
      value from vector data
7. Calculate flexure variation from
      each sensor
8. Apply variations to fingers using
      local coordinate system
9. (If active) Apply camera movement
      mapped by glove gesture
```

Figure 14: Glove data pseudo code.

```
1. Get orientation data from 3D Visor
      gyroscope
2. Calculate eMaginBB execution time
3. Attenuate high-frequency noise
      with low-pass filter
4. Data insertion on a vector with a
      pre-defined size
5. Eliminate oldest value if vector
      full
6. Calculate an arithmetic average
      value from vector data
7. Calculate new orientation
8. Apply new orientation to active
      game camera using local
      coordinate system
```

Figure 15: eMagin gyroscope pseudo code.

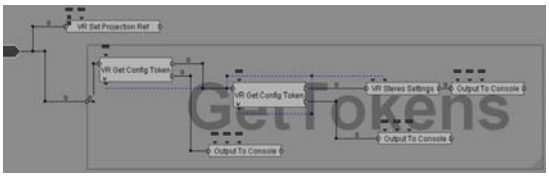

Figure 16: Stereoscopy configuration script.

4.1 PongVR

This game's main goal is to simulate a real table tennis game. A player uses tracking information to position and orient a racket in the 3D virtual world: a 3D Visor is used to visualize the game and orient the player's camera using its internal gyroscope. A 5DT glove is put on the player's left hand with the purpose of moving the camera with a set of predefined gestures. The game's physics simulation is achieved through Virtools Physics pack. A hit combo system is also created to enhance the player's

gaming experience. Video demonstrations of this game are publicly available in the youtube [1]..

PongVR offers the possibility to configure all sensors sensibility from within the game, through the Options Menu accessible in the Start Menu or pressing 'P' during the game. Three difficulty modes Easy, Medium and Hard are programmed, which defines the opponent AI.

4.2 PianoVR

In this game the player controls a pair of virtual hands in order to play a piano. Two tracker receivers are used, one to each hand, so the player can move and orient both virtual hands.

Besides the tracker data, a pair of 5DT gloves are put on and their sensors information are used to transform a bone system within each of the virtual hands. Every one of these bones has a weight associated to determine how the hands mesh will be modified. The process of creating a bone system, associating this info into a mesh and transform the bones is called skinning (Virtools, 2006).

Virtools Physics Pack wasn't used in PianoVR. The interaction between hands and piano were developed specifically to this game. Video demos of this game may be seen in youtube [2].

The sensibility of all the glove sensors and tracker may be changed from inside the game using the Options Menu available from the Start Menu or pressing ´P ´when in game.

Like PongVR, this game makes use of the 3D Visor to view the game with active stereoscopy and to orient the player's cam.

In order to increase gaming experience, a series of pre-determined music's are placed into the game and the player's objective is to successfully play each one of these songs within a limited time. Three difficulty modes are given to the player, Easy, Medium and Hard. They define how much time the player has to perform each one of the predefined music's.

5 PERFORMANCE MEASUREMENTS

A series of tools were created to know how well the immersive devices behave inside Virtools. A new Manager, Time Manager, along with a new behavioural Building Block, named Get_Precise_Time, were developed to access the internal CPU clock and used to perform time measurements with a resolution of µ seconds.

The time variable supplied by the BB is the time elapsed since the activation of the Time Manager from within Virtools, so the execution time of GloveBlock or TrackerBB is determined by using a Get_Precise_Time BB before and after those BB. Both Glove Manager and Tracker Manager also calculate their internal latency time. Combining this information with the one given by Get_Precise_Time BB, the overall time since the data is requested to the devices until it is used inside Virtools can be accurately found.

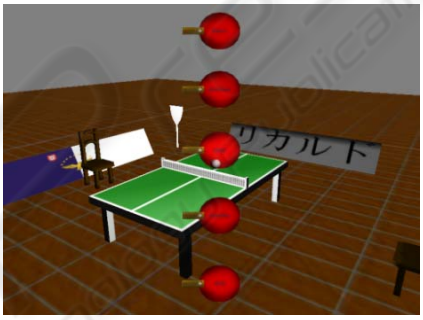

Figure 17: PongVR Main Menu.

Figure 18: PongVR Combo System.

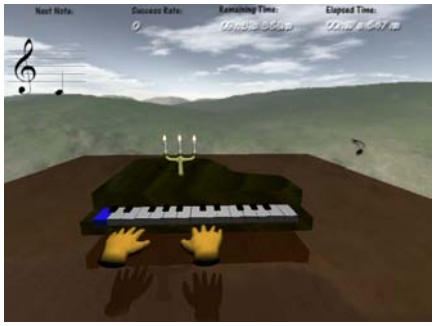

Figure 19: PianoVR in game footage.

[1] http://youtube.com/watch?v=wEDRyzol-J8
http://youtube.com/watch?v=futpUwc2A0A
[2] http://youtube.com/watch?v=PSGD9bRWvWw
http://youtube.com/watch?v=EFJttobtFrU

Figure 20: Finger-piano key collision.

Both games supply latency time information to the player by pressing '1' key. A population of 100 samples is captured to calculate average values. The table bellow shows time results obtained executing a composition with just the devices BB, and with both games PianoVR and PongVR. The measurements are made in a laptop with a 1.8GHz Core Duo processor, 1024 of RAM and an ATI X1600 Mobility Radeon Graphics Card.

PianoVR has a total of 9786 behaviours, 102 3D objects with a total of 7233 faces and makes use of 5.61 MB of Video Ram and PongVR has a total of 625 behaviours, with 48 3D objects, 9063 faces and makes use of 4.2 MB of Video Ram. This makes PianoVR the worst case scenario and it will be used as a reference.

Magnetic Tracker **-** The tracker has latency time value of approximately 34ms which is bigger than expected since given by (Polhemus, 2005), that sums a total of 20.67 ms. The time differences is caused by the filters internal latency time to calculate new values of position and orientation to its receptors and the fact of the application running inside Virtools environment also slightly increases the latency time.

Data Gloves **-** The gloves overall time sums to a total of 123µs. Information regarding the sampling frequency (5ms) is provided in (5DT, 2006) for the 5 sensor model but no information is available regarding the internal latency time. With such a low value of overall time means that only USB communication latency is taken into account and the glove device provides information even without having new flexure data available.

Glasses Gyroscope **-** For the 3D Visor model, a value of 4ms device latency time and 125Hz sampling rate is provided in which gives a total of 12ms total latency time. The discrepancy between the official value and the one obtained through the eMagin Manager is due to the polling time of 30 ms by eMaginBB pIn, which is also taken into account. Also the fact of the application is running inside Virtools environment slightly increases the latency time.

6 USER TESTS

Some user tests were made in order to evaluate how well the PongVR and PianoVR interact with a player and if they have a good immersive experience.

Each one of the users first learn how each device operates and then plays both games. In the end a poll is made in which the player have to give a score from 1 to 10 to some questions.

The global evaluation for both games is quite good. All users have felt a good immersive experience while playing both games, and had a positive response in how the devices features were integrated into the games. They had also a successfully augmented depth perception caused by active stereoscopy from 3D Visor. However they felt that both games, especially PongVR, had a high learning curve mostly caused by the magnetic tracker behaviour.

New features were added into both games after the user tests namely:

Possibility to change the eMagin Z800 3D Visor polling time to increase/decrease its gyroscope sensibility.

Real time calibration of stereoscopic camera field of view and distance to the convergence plane so that a user may adjust these values with the purpose of enhancing depth perception

Table 1: Devices Performance.

	Empty Scene	PianoVR	PongVR
GloveBlock (µs)	8	345	56
TrackerBB (µs)	8	970	45
eMaginBB (µs)	41	80	200
Glove Manager (µs)	123	123	123
Tracker Manager One Receiver (µs)	22000	-	22000
Tracker Manager Two Receivers (µs)	33000	33000	-
eMagin Manager (µs)	54000	54000	54000

Table 2: PongVR Poll questions.

PongVR Total	Answer [1-10]
Tracker sensitivity	8,5
Tracker readyness	8,8
Trackers features Integration	9,0
3D Visors gyroscope sensitivity	6,8
3D Visors gyroscope readyness	6,8
3D Visors features Integration	8
Depth perception	7
Immersive experience	7.5
Gameplay	8,2
Game difficulty	8,3
Game global evaluation	7,8

Table 3: PianoVR Poll questions.

PianoVR Total	Answer [1-10]
Tracker sensitivity	9,2
Tracker readyness	8, 7
Trackers features Integration	9,2
Gloves sensitivity	8,2
Gloves readyness	8, 3
Gloves features integration	8,5
3D Visors gyroscope sensitivity	6,8
3D Visors features Integration	7
3D Visors features Integration	7,8
Depth perception	6,8
Immersive experience	7,7
Gameplay	8,5
Game difficulty	7,5
Game global evaluation	8,2

7 CONCLUSIONS

The goal of this work was to use professional developing tools to create games using state of the art immersive devices. Virtools plugins and internal scripts were developed in a modular way, so that they can be reused in other contexts and applications. The focus of this work applications were game development, but an industrial application might have been created instead, e.g. a virtual car assembly line where the user has a virtual storage and can grab, translate and rotate parts to assemble a car prototype.

Game design was seriously taken into account while developing each one of these games. The creation of an appealing interface, which gives all the necessary information to the player without distracting him from the game objective, was a constant preoccupation. Both games had a modular architecture with different threads handling game logic, physics, etc.

However the devices have several issues that decrease gameplay. Magnetic tracking device has a limited range of action and relatively high latency times. Both the data gloves, 3D Visor and Tracker are connected via USB cable into the PC so the player has a confined area of movement restricted by the several cables.

Nevertheless, both games PianoVR and PongVR were successfully created allowing the player to have a very immersive experience while playing not being limited to traditional input/output devices.

REFERENCES

ELSPA – Entertainment and Leisure Software Publishers Association, 2003. The Cultural Life of Computer and Video Games: A Cross Industry Study. ELSPA Publications, Worcestershire, UK.

Goebel, M., Hirose, M., Rosenblum, L., 2001: Today's VR. *IEEE Computer Graphics and Applications, November/December 2001 21(6) 22-24.*

Hinckley, K. R.J.K. Jacob, and C. Ware, 2004. Input/output Devices and Interaction Techniques," in *The Computer Science Handbook, Second Edition*, ed. by A.B. Tucker, pp. 20.1-20.32, Chapman and Hall/CRC Press.

Polhemus, 2005. 3Space™ Fastrak® User's Manual. Polhemus Inc.

5DT, 2004. 5DT Data Glove Ultra Series User's Manual, 5DT Corp.

eMagin, 2005. eMagin z800 3DVisor User Guide, eMagin Corporation.

Mc Carthy, C., Callele, D., Krupa, F., 2006. *Virtools User Guide*, Dassault Systemes.

Nahon, D., Philippon, N., Kaci L., Kuntz, S., 2005. *Virtools™ Behaviour Libraries - VR Library/VR*, Dassault Systemes.

Virtools, 2006. Virtools™ 4.0 Online Reference, Dassault Systemes.

5DT, 2006. 5DT Data Glove Series, 5DT Virtual Reality for the Real World (http://www.cybermind.nl/Info/ 5DTDataGloveSeries.pdf).