The Utilization of Low Cost Gaming Hardware in Conventional Simulation

Peter Smith, Lee Sciarini
Institute for Simulation and Training
Orlando, Fl
psmith@ist.ucf.edu, lsciarin@ist.ucf.edu

Denise Nicholson
Institute for Simulation and Training
Orlando, Fl
dnichols@ist.ucf.edu

ABSTRACT

The current convergence between conventional simulation training and game based training is blurring the lines that once defined each industry. The simulation industry has already begun to assimilate low cost gaming middleware for graphics, artificial intelligence, and physics. With the unequivocal success of the integration of game based software solutions into conventional training , the next logical step is the utilization of low cost gaming console hardware within the tool chain.

The most recent generation of gaming console hardware is ripe with potential uses within simulation and training. The three major game console manufacturers, Microsoft, Nintendo, and Sony, have each provided open tools that allow direct access to their various hardware solutions. The potential of these systems envelop a broad range of uses from deploying fully immersive 3D simulations and performing computationally intensive multithreaded tasks to revolutionary paradigms for user interaction.

Determining the appropriate solution for a given situation can be a daunting task for uninitiated developers, as each solution is unique to its hardware. Given this, training objectives and requirements must be mapped to the full range of fidelity and capabilities offered by each solution. A recent study to determine the utility of low cost gaming hardware in the Deployable Virtual Training Environment (DVTE) was conducted by a multidisciplinary team from the Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) Lab at the University of Central Florida's Institute for Simulation and Training. This work discusses the initial findings from the study and provides guidelines for employing the latest generation of gaming hardware within PC Simulation and Training.

ABOUT THE AUTHORS

Peter Smith is currently employed as Visiting Research Faculty at the University of Central Florida, in Orlando Fl. He works in the Active Lab as the resident games researcher, and is also concurrently pursuing his PhD. in Modeling and Simulation. His previous employment as a Serious Games and Simulation Engineer at the Navy's NETC Experimentation Lab exposed him to the field of Serious Games. Peter is also involved as a volunteer with the Serious Games Initiative, and is an avid blogger on The Second Life Insider.

Lee W. Sciarini is earning a Ph.D. in Modeling and Simulation under the supervision of Dr. Denise Nicholson in the Applied Cognition and Training in Immersive Virtual Environments Laboratory (ACTIVE) at the University of Central Florida. For the past 2 years, his work has been focused on the training effectiveness of systems, including The Multi-platform Operational Team Training Immersive Virtual Environment (MOT2IVE) system, the Combined Arms Planning Tool (CAPT), JSAF and the Dismounted Infantry Virtual After Action Review System (DIVAARS) after action review system. His research interests include simulator training effectiveness, human-systems interaction and augmented cognition.

Denise Nicholson is the Director of the Applied Cognition and Training in Immersive Virtual Environments Laboratory (http://active.ist.ucf.edu/) at the University of Central Florida's Institute for Simulation and Training. She is also a faculty member in UCF's Modeling and Simulation Graduate Program, a Research Scientist in the College of Optics and Photonics/CREOL, and a Certified Modeling and Simulation Professional. Dr. Nicholson's research focus on human systems modeling, simulation and training includes virtual reality, human–agent collaboration, and adaptive human systems technologies for Department of Defense and Duel-Use applications. She has authored/coauthored more than 40 technical publications and given numerous invited lectures/presentations.

The Utilization of Low Cost Gaming Hardware in Conventional Simulation

Peter Smith, Lee Sciarini
Institute for Simulation and Training
Orlando, Fl
psmith@ist.ucf.edu, lsciarin@ist.ucf.edu

Denise Nicholson
Institute for Simulation and Training
Orlando, Fl
dnichols@ist.ucf.edu

Introduction

The current generation of gaming console platforms including the Xbox 360, Playstation 3 and Wii, all offer different levels of user customization and the ability to repurpose hardware for use in Modeling and Simulation applications. No previous generation of gaming hardware has provided appropriate interfaces at a consumer level price. While the API's that are being provided are aimed at consumer applications, the military is interested in the feasibility of adapting consoles for deploying simulation based training (Smith, 2006).

The Applied Cognition and Training in Immersive Virtual Environments (ACTIVE) Lab at the Institute for Simulation and Training (IST) is currently engaged in a number of training projects developed on game based platforms. One of the most interesting candidates for applying video game console platforms is within the Deployable Virtual Training Environment (DVTE).

However, it is not immediately clear which gaming systems will provide the best interfaces for the DVTE platform. Unfortunately none of the consoles provide a single solution that would meet all of the needs of a modern simulation system such as DVTE. Exploring the strengths and weaknesses of each platform is crucial before planning any integration of gaming hardware into a simulation pipeline.

DVTE Platform Overview

DVTE is the latest undertaking of the Virtual Technologies and Environments (VirTE) program. Its primary use is to train Fire Support Teams (FiST) in the planning and execution of calls for indirect fire (artillery and mortar) and close air support (CAS). At its core it primarily runs on top of the Delta3D Open Source Game and Simulation Engine, however, certain applications within the DVTE platform do not. For example it can interface with Joint Semi-Automated Forces (JSAF), Dismounted Infantry Virtual After Action Review System (DIVAARS), and other DIS/HLA applications. That said, the main simulation interface is a 3D view that can be displayed in multiple fidelity environments.

Console Hardware in Simulation Overview

Game Console hardware can provide output at equivalent quality to a high end PC at the cost of a low end PC. The current generation consoles range in price from $250.00 USD to $600.00 USD. While their prices, and ultimately their functionality, vary greatly they all cost less than a standard PC, no less a gaming hardware equipped one. From a financial perspective these platforms warrant further exploration.

Xbox 360 Overview

The Xbox 360 is the latest console from Microsoft. It included a 3 Core, dual threaded processor from IBM. When superficially compared to the Playstation 3 specifications the Xbox 360 may seem underpowered; but it is a very capable machine.

Table 1. Xbox 360 System Specs (Microsoft, 2007)

CPU Cores	3 Dual Threaded Cores
CPU Speed	3.2 GHz
GPU	500 MHz ATI
Memory	512 MB Unified
Storage	Detachable HDD
I/O	USB/Wireless Controllers

With a head start on the market of almost a year, its games are outshining the Playstation 3's, mostly due to developer familiarity with the platform.

Playstation 3 Overview

The Playstation 3 is undoubtedly the most powerful of the new generation of consoles. It provides a unique nine core architecture that can process data faster than most computers available today.

Table 2. Playstation 3 System Specs (Sony, 2007)

CPU Cores	1 PPE 8 SPE
CPU Speed	3.2 GHz
GPU	550MHz RSX
Memory	256 Main, 256 VRAM
Storage	Detachable HDD
I/O	Bluetooth Controllers

The actual hardware is divided into 1 PPE or Power Processing Element that has power similar to a Power Mac G5 and 8 SPEs of Synergistic Processing Units. These processors alone have their own L2 Cache and individually have about as much power as the Playstation 2.

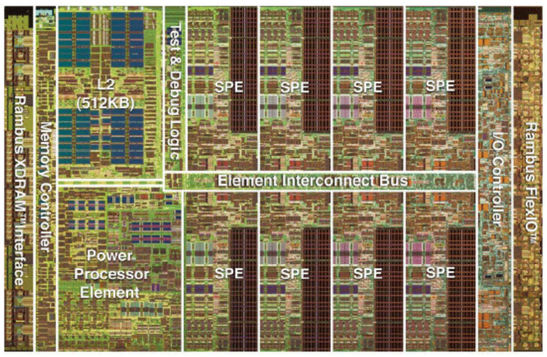
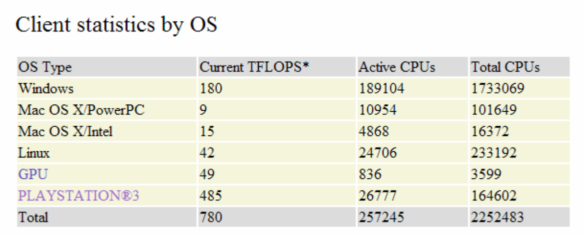


Figure 1. Cell Processor (MIT, 2007)

Due to inconsistencies in manufacturing the Playstation 3 code never uses more than 7 of the SPEs at once.

In order to gain an understanding of how powerful the Playstation 3 can be one might consider the Folding @ Home project; which uses distributed computing to perform protein folding experiments. They have developed a client for the Playstation 3 as well as the PC and Mac.



Client statistics by OS

OS Type	Current TFLOPS*	Active CPUs	Total CPUs
Windows	180	189104	1733069
Mac OS X/PowerPC	9	10954	101649
Mac OS X/Intel	15	4868	16372
Linux	42	24706	233192
GPU	49	836	3599
PLAYSTATION®3	485	26777	164602
Total	780	257245	2252483

Figure 2. Folding @ Home Stats (Folding @ Home, 2007)

As can be seen in figure 2, while the nearly 200,000 Windows PCs connected to the Folding @ Home network produce 180 Terra Flops of data crunching power, 485 Terra Flops can be provided by 6 times fewer networked Playstation 3s.

Wii Overview

The Wii is hard to categorize in this generation of consoles. Its hardware is considerably underpowered, providing what only amounts to an upgrade to Nintendo's previous generation hardware.

Table 3. Wii System Specs (Nintendo, 2006)

CPU Cores	IBM Single Core
CPU Speed	729MHz
GPU	243MHz
Memory	88MB Unified
Storage	Flash Memory Drive
I/O	IR/Bluetooth Controllers

However, the source of interest for simulation in the Wii is not in its main console hardware, but in its innovative controller.

Figure 3. Nintendo WiiMote (Nintendo, 2006)

The WiiMote, shown in figure 3, can track a players hand motions using a combination of IR, and Accelerometers. It is also a standard Bluetooth device that can be interfaced with a PC.

CONSOLE HARDWARE CHARACTERISTICS

As stated earlier the financial prospect of using game console hardware in simulation makes sense. This is true, until the development pricing model gets involved. Previous successful experiments with console hardware such as the Army's Full Spectrum Warrior, developed with Pandemic Studios was criticized for costing the government and estimated $5 million dollars to build on the original Xbox. (Adair, 2005) With expected costs for the current generation to be around $20 million dollars per game the prospect of more conventional crossover games is slimming (BBC 2005).

While the development costs for commercial games is rising, the consumer is demanding the ability to create their own games. Console manufacturers have responded to these needs by opening up their systems to individuals to develop their own games, however, each is adopting their own strategy. Each different strategy affects the usefulness of adopting their hardware in significant ways. For example, the Xbox has an elegant API, but no direct access to the hardware. The Playstation provides the ability to install Linux, but does not provide an easy to use API. The Wii, on the other hand does not allow for any programming at all, but its innovative controller can be interfaced with a PC.

The most important characteristics that need to be evaluated with these attempts to integrate gaming hardware into simulation platforms are access to hardware, robust APIs, and repurpose-able hardware.

Table 4. Next Gen Consoles Vs. Important Characteristics for Training

	Characteristics		
	Access to Hardware	**Robust API**	**Repurposible Hardware**
Xbox 360	Controlled Interface	XNA and Freely Available DevEnv	Controller Only
PS3	Linux Installable	GCC and Open Source API	Controller and System
Wii	Controller Only	None	Controller with Immersive Interface

Table 4 compares each console on the characteristics defined above. The following sections will discuss each characteristic, access to hardware, robust APIs, and repurpose-able hardware, fully.

Access to Hardware

One important characteristic of gaming consoles is the available level of access to the hardware. Historically, console hardware has remained a closed system that the users are explicitly blocked from accessing. However, as consumers are becoming more sophisticated users and manipulators of technology, they have imposed their own hacks (unauthorized modification resulting in access to otherwise unavailable features) upon the last generation of consoles to allow access to the hardware without the explicit approval of console manufacturing companies.

In reaction to console hacking the manufacturers have not denounced the user hacks, but instead have embraced this segment of the market and provided them with tools to hack in a sanctioned way. This allows users to have the access they want while providing alternatives to the hacking of consoles for play of pirated software. These initiatives by console manufacturers have resulted in unprecedented access to the hardware.

Unfortunately, most of the solutions provided still do not allow for writing low level code. For example, Microsoft has created XNA; a managed code layer that shields the user from low level hardware access. The Nintendo Wii provides no official access to hardware, although the WiiCade (WiiCade, 2007) provides Flash games optimized to play with the WiiMote through the Wii browser.

Surprisingly, Sony's Playstaion 3 is the true standout in hardware access. In an uncharacteristic move by the historically proprietary company, Sony has allowed the user to install Linux onto the Playstation 3 straight out of the box. They even worked closely with Yellow Dog Linux to provide a version that supports the multiple core architecture of the Playstation 3.

The ability to run Linux allows the user to recompile existing Linux code to run on the Playstation 3. They can even modify that code to take advantage of the core processor. Being able to run HLA networking code is a highly desirable feature of any console based simulation, a feature that Microsoft's XNA architecture does not currently support. This feature will be discussed further in the Robust API section below.

While being able to run Linux initially seems like the perfect solution to console access; there are a few caveats that hold the Playstation 3 back as the console of choice for Simulation. First, Sony has reserved one core for running the original Playstation 3 OS. Combining this with the fact that Sony already restricts access to only seven cores, only six cores are fully available. Additionally, the system is currently without a graphics driver for the GPU, severely handicapping the system in graphics output capabilities.

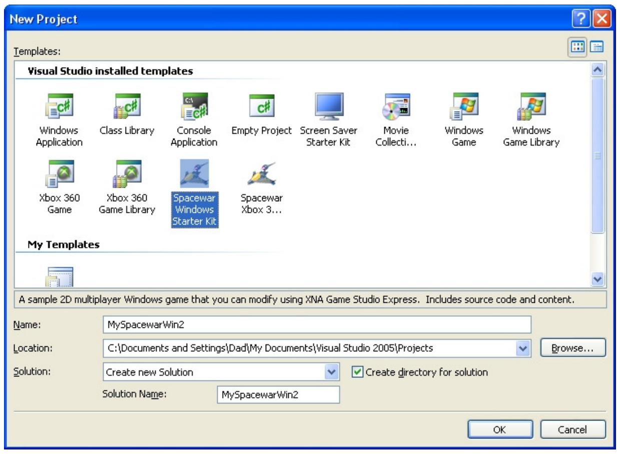
Robust API's

Another import characteristic to consider when selecting a console hardware to leverage in a simulation application is the availability of robust programming API's. To truly utilize console hardware a developer will eventually need to write some code. The availability of resources and the ease at which these tasks can be learned can make the difference between doing a simple conversion and requiring employees with unique skill sets targeted at exposing the underlying hardware.

It is important to remember that Nintendo does not provide any API's for its hardware. However, the other two major consoles do.

The most robust and easy to learn API for console development is Microsoft's XNA developer's kit. XNA is a library that sits on top of C# and allows game code to be written and compiled in C#. It is freely available and only requires the C# Express version of the Visual Studio environment.

Developers can create a new project upon launching C# Express and may select to make a game using XNA for Windows or the Xbox 360. To switch between platforms the developer need only recompile.



A sample 2D multiplayer Windows game that you can modify using XNA Game Studio Express. Includes source code and content.

Name: MySpacewarWin2

Location: C:\Documents and Settings\Dad\My Documents\Visual Studio 2005\Projects

Solution: Create new Solution [✓ Create directory for solution]

Solution Name: MySpacewarWin2

Figure 4. XNA Starter Kit Section Menu (Microsoft, 2007)

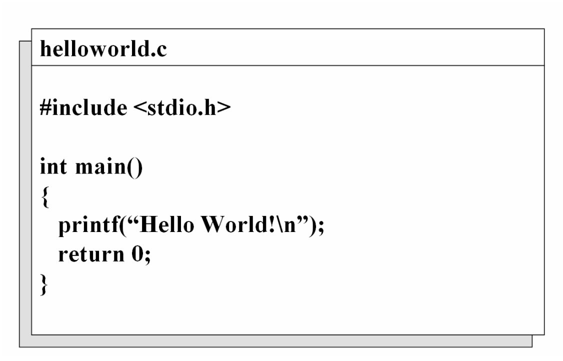
There are also a number of starter kits that include full code for anything from a 2D side scroller game to a full 1080p racing game. While none of the starter kits explicitly provide a simulation environment; moving from a starter kit to a working simulation application could be done in short amount of time.

Figure 5. Racing Game Starter Kit (Microsoft, 2007)

XNA keeps all of the programming at a managed code level, which means the developer never has to consider the underlying hardware to accomplish favorable results. This maintains the API's ease in learning and utilization. This inability to truly get into the hardware of the console through the API, however, is also the greatest drawback. A developer using XNA can not get true access to the power of the Xbox 360.

When comparing this to the Playstation 3 and its Linux operating system, it is hard to settle for such a hindered level of control. Unfortunately the API on the Playstation 3 is much less robust. Even the commercial version of the Playstation 3 development kit provides a series of open source or freely available API's including PSGL (OpenGl ES modified to use Cg Shaders instead of GLSL Shaders), Collada model file support, OpenMax, OpenVG, and a handful of proprietary tools from SN Systems.

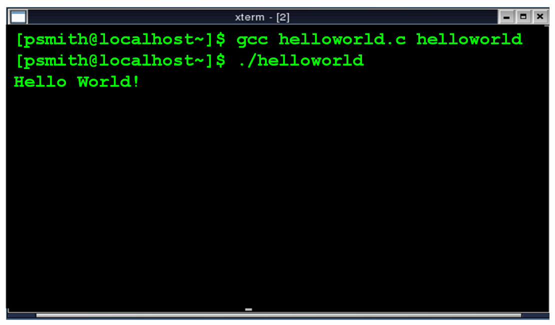
While all of these APIs are available to the user, the Cell architecture adds a new level of complexity to the code writing task. The standard Hello World application has become iconic as a first step to developing on any system.



```
helloworld.c

#include <stdio.h>

int main()
{
   printf("Hello World!\n");
   return 0;
}
```

Figure 6. example helloworld.c

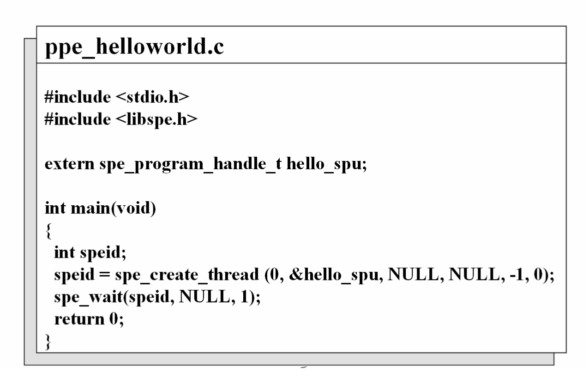
The code in standard C/C++ is very straight forward and easy to read and understand. Compiling the code using GCC and running it is just as easy.



```
[psmith@localhost~]$ gcc helloworld.c helloworld
[psmith@localhost~]$ ./helloworld
Hello World!
```

Figure 7. example Hello World terminal window

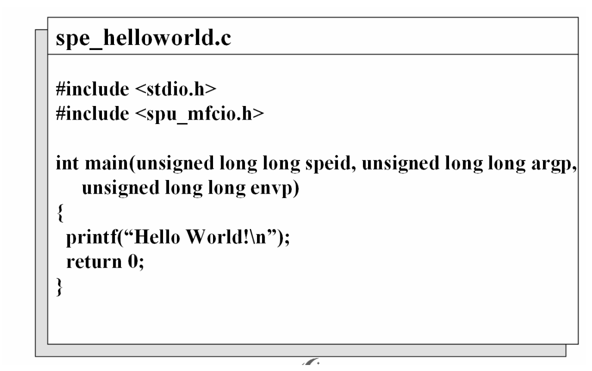
Of course this is in a single Core system. When utilizing an additional core complexity is added to the code. Consider the same code written for a Cell processor. There are some additional libraries needed to develop the code. The critical part in this code is in the call spe_create_thread. This code creates a new thread on a new Core.



```
ppe_helloworld.c

#include <stdio.h>
#include <libspe.h>

extern spe_program_handle_t hello_spu;

int main(void)
{
  int speid;
  speid = spe_create_thread (0, &hello_spu, NULL, NULL, -1, 0);
  spe_wait(speid, NULL, 1);
  return 0;
}
```

Figure 8. ppe_helloworld.c

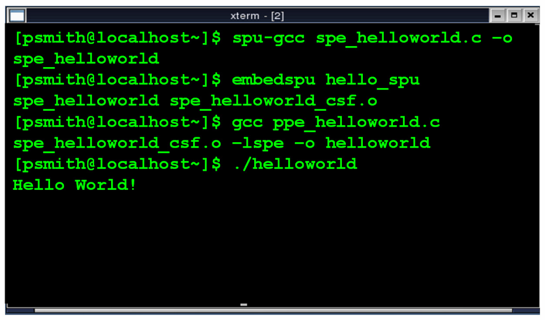
The code that runs in this thread is defined to do the same Hello World functionality and is launched from the spe_create_thread line above. The extern defined in the ppe_hellowworld.c file corresponds to this code and calls it hello_spu.



```
spe_helloworld.c

#include <stdio.h>
#include <spu_mfcio.h>

int main(unsigned long long speid, unsigned long long argp,
    unsigned long long envp)
{
 printf("Hello World!\n");
 return 0;
}
```

Figure 9. spe_helloworld.c

You may have noticed that the spe_helloworld.c code does not contain any reference to hello_spu. This is actually defined at compile time. Thus, this transforms the standard process of compiling from one step to a 3 step process. This is shown in the console terminal window below.



```
[psmith@localhost~]$ spu-gcc spe_helloworld.c -o
spe_helloworld
[psmith@localhost~]$ embedspu hello_spu
spe_helloworld spe_helloworld_csf.o
[psmith@localhost~]$ gcc ppe_helloworld.c
spe_helloworld_csf.o -lspe -o helloworld
[psmith@localhost~]$ ./helloworld
Hello World!
```

Figure 10. terminal window on PS3

First the spe application must be compiled using a special version of the GCC. It is followed by embedding the code into the hello_spu. Finally the application is built with the embedded code included.

In this case it is a trivial example, but more complex examples would only require additional compile steps, as each program destined to be run on an SPE needs to be compiled separately.

This example demonstrates that there is a trade off between how easy an API is to use and how much control you have over the hardware. The XNA starter kits can jump start code with little or no knowledge of what is happening on the Xbox. While the Playstation 3 code could take much more time to get started, it provides Core level access to what is happening on the CPU. Therefore, each has its own set of advantages and disadvantages.

Repurpose-able Hardware

While possibly the least exciting use of console hardware, the idea of repurpose-able hardware yields the most success stories for console hardware. In a recent test of the Future Combat System (FCS) platform; pictures revealed that the Army has employed an Xbox 360 controller to operate the Red Owl Sniper Robot.

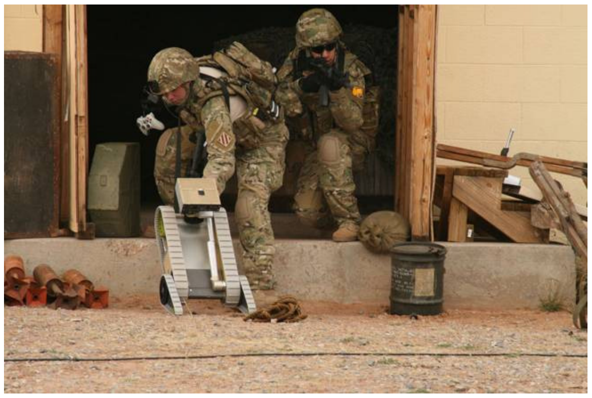


Figure 11. FCS Test using Xbox 360 Controller (Smith, 2007)

The Xbox controller provides a standard USB PC interface. They connect to PCs as standard game pads and provide a consistent interface between the PC and the Xbox 360 console. In applications where a controller interface makes sense, using the Xbox 360 controller is a huge win for console hardware utilization.

The Xbox 360 controller, however, is the least extraordinary controller in the current generation of consoles. Conversely, the WiiMote has defined a new paradigm for how people interact with gaming consoles. It uses a combination of IR and Accelerometers to track the actual movement of the player's hands. Additionally the WiiMote operates as a standard Bluetooth device that can be connected to any PC. The ACTIVE Lab has developed an API for integrating this unique controler into the simulation pipeline and it now serves as a low fidelity replacement for the Intersense Tracker used to track players in high fidelity simulations.

Figure 12. WiiMote Vs. Intersense (Smith, 2007)

While the cost of an individual WiiMote averages $40.00 USD, the cost of an Intersense Tracker averages $4500.00 USD. Therefore, the image below (Figure 13) is a more accurate representation of the price relationship between devices than the figure above (Figure 12).

Figure 13. Wiimote Vs. Intersense with cost considered (Smith, 2007)

It is the intention of the ACTIVE Lab to publish the interface code as Open Source in the near future, so that it can serve as the standard for interfacing between the WiiMote and the PC.

CONCLUSIONS

While the ideal conclusion would be that there is one standout console that will solve all of the simulation industry's needs, this simply is not the case. The console market is just beginning to open its mind and hardware to the idea of allowing external developments outside the mainstream consumer driven games market they have historically tapped as a source of revenue.

These first strides are showing that different open development models can be applied to consoles to create immediate and useful results. While the Xbox will not be the perfect platform for DVTE due to its inability to run the complex networking code required; it is a great platform for non-networked 3D simulations.

Conversely, the Playstation 3 will not be the home of a 3D simulation; although it could run JSAF or other non visual networked simulations. The ACTIVE Lab is also exploring the Playstation 3 platform for robotics simulation by taking advantage of the high level of computing power available for the price.

Even more interesting is the fact that the military is already using Xbox 360 controllers, and the new levels of control available from the Wii are also looming on the horizon. While these victories are small in comparison to industry as a whole, they are important first steps in the utilization of console hardware long term. As long as these first successes are highly praised and publicized, manufacturers of the next generation gaming consoles will be compelled to provide even more openness and support to non-conventional game development. As this occurs it will transform how consoles are viewed by the consumer market and the simulation and training community as well.

ACKNOWLEDGEMENTS

Thanks to the ACTIVE Lab and its entire staff.

REFERENCES

Adair, B. (Feb, 2005). Did the Army get out gamed? Times Washington. Retrieved June 2007 from http://sptimes.com/2005/02/20/Worldandnation/Did_the_Army_get_out_.shtml

BBC News (17 Nov 2005). Cost of making games set to soar. Retrieved June 2007 From http://news.bbc.co.uk/1/hi/technology/4442346.stm

Microsoft (2007). XNA Team Blog. Retrieved June 2007 from: http://blogs.msdn.com/xna/

MIT. (2007). Multicore Programming Primer: PS3 Cell Programming. Massachusetts. Retrieved March 2007, from http://cag.csail.mit.edu/ps3/index.shtml

Nintendo (2006). Nintendo Wii Homepage. Retrieved June 2007 from http://www.wii.com

Smith, P. A. & Gruhl, R. (2007). Serious Games Next-Gen. Serious Games Summit. Game Developers Conference 2007. San Francisco.

Smith, R. (2006). The Challenge Map for Serious Games. Serious Games Summit DC 2006. Washington D.C.

Sony (2007). Playstation 3 Homepage. Retrieved June 2007 from: http://www.playstation.com

WiiCade (2007). Wiicade. Retrieved June 2007 from http://www.wiicade.com