In the 2nd International Conference on Life System Modelling and Simulation (LSMS'07) Shanghai, China, September 14-17 2007

Modelling Pervasive Environments Using Bespoke & Commercial Game-Based Simulators

Marc Davies[1], Vic Callaghan[1] & Liping Shen[2]
[1]Digital Lifestyles Centre, Essex University, UK
[2]eLearning Lab, Shanghai Jiao Tong University, China
midavi@essex.ac.uk, vic@essex.ac.uk, lpshen@sjtu.edu.cn

Abstract. This paper details an ongoing investigation, linking intelligent buildings and computer game technology. Intelligent buildings are containers for life-sized organisms (people, pets etc), sustaining ecologies that evolve and interact in a symbiotic way with the technological infrastructure, which includes intelligent agents. This work explores how computer games software can be used to create a simulation tool for the development of new ubiquitous agent programs and environments. We report on our experiences of adapting a retail package, (Electronic Arts' *Sims*) and building our own bespoke simulator. We use a table to compare and summarise the strengths and weaknesses of each approach; the general conclusion being that there are significant benefits to be gained from adapting commercial games packages for use as professional simulation tools. Finally, we conclude the paper by describing our plans to apply this work to the development of a mixed-reality eLearning application.

1 Introduction

1.1 Why Use a Simulator over a Real-World Test-bed?

Technology is assuming an ever increasing role within the environments in which we live and work [4]. Developing intelligent embedded-agents and pervasive computing environments can be a costly process. When testing agents in a real-world environment, researchers can only perform experiments in real-time (e.g. with each iteration taking days) and often have little control over natural elements, (e.g. sunlight). The reasons for this are that agents model and learn behaviours by monitoring people, as they go about their everyday activities [1]. This makes it practically impossible to repeat tests under identical conditions. When developing a new pervasive agent, current researchers inevitably need a physical device, and/or environment to test their program.

Building a simulator from scratch is a formidable task, requiring a considerable programming effort to create realistic graphical environments and user-behaviours (especially interactive behaviours). We hypothesized that a Pervasive Environment Simulator, (PES) based on computer game technology, could take advantage of the existing high-detail graphics physics and some artificial intelligence techniques used in modern games, to provide a high-quality model of a real-world test-bed. Additionally we hypothesized that creating a PES by modifying a computer game would be easier and produce a higher quality simulation than a bespoke system.

1.2 Existing Projects

Computer games and related technologies are already being used for researching non-entertainment applications by various private and public organizations around the world. The Entertainment Technology Center at Carnegie Mellon University has run a project using their 'Alice' graphics engine, to teach artists how to program virtual worlds [11]; In another example, the U.S. 'Darwars' training program uses a simulated environment based on the Unreal Tournament game engine, to produce foreign landscapes where soldiers can interact with virtual inhabitants allowing them to acquire language skills [18]. Computer Science skills are taught by allowing participants to design their own computer games, matching the expectations of younger generations, who are unimpressed by simple visualizations [13].

Bespoke simulators are already being used to model household environments. The 'MavHome' Project [5] consisted of an intelligent living environment, with certain components, (e.g. window blinds) controllable from by a simulator created for the project [5]. Simulators are also being used to augment telemetry from sensors in real-world locations, improving the reliability of readings and allowing other operations that would otherwise be impossible. For example a security system that augments a camera-feed to allow an observer to 'see-through' solid objects, by rendering the hidden space in the simulator [14]. Additionally, simulators can provide training tools for complex tools and vehicles, e.g. flying aircraft, manoeuvring underwater robots etc [3]. Aircraft simulators have been converted into several popular computer game titles, most notably the 'Microsoft Flight Simulator' series, useful to prospective pilots learning to fly [12] and of high entertainment value to gamers around the world. This provides encouragement for this project. As a simulator can be converted into a commercial game, the process should be reversible, with games modified into simulators for research and development applications?

1.3 Bespoke Vs. Game-Based Simulators

As part of our research two simulators were developed; a) A completely bespoke system, written using the Java programming language. b) A simulator created by modifying an off-the-shelf copy of a popular computer game. Both programs modelled the iDorm2, at the University of Essex. A full sized two-bedroom apartment, constructed to be a pervasive computing test-bed, featuring hollow walls and ceilings fitted with a myriad of embedded-computer based technology [2] [8].

Figs. 1-4. Views of the University of Essex iDorm test-bed

2 Simulator Design

2.1 Bespoke PES

Design Rationale. A two-dimensional PES, (2D-PES), aimed to provide a benchmark, for comparison with the game-based PES. Additionally by creating a bespoke system we were able to further our understanding of the program architecture required. This knowledge was used when implementing the game-based 3D-PES, letting us identify the components of the original program requiring modification. It was important to identify how pervasive devices and features could be simulated, later assess which could be incorporated into a game-based PES.

Simulator Architecture. In more technical terms the 2D-PES consisted of four Java programs, linked by socket communication; a) Two-programs operating the actual simulator. b) A central server program. c) A third-party agent program, also written using Java. Threads were used to handle time-delays and loops in program code.

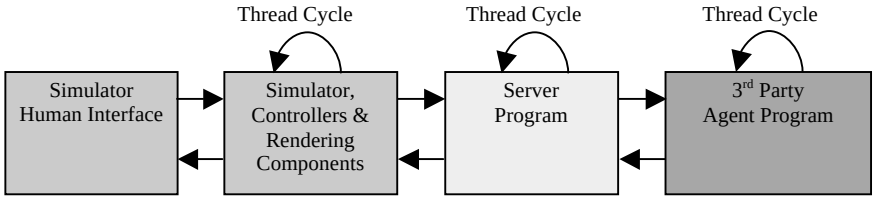


Fig. 5. Bespoke 2D-PES Architecture

The central server regulated the exchange of all data between the simulator and the third-party agent program. Readings for sensors in the environment were transmitted from the simulator to the agent program on each thread cycle. A list of settings was also received by the simulator, from the agent program, containing states pervasive objects needed setting to by the simulator. The agent determined settings by analysing sensor data sent on the previous thread cycle.

Fig. 6. The Bespoke 2D-PES

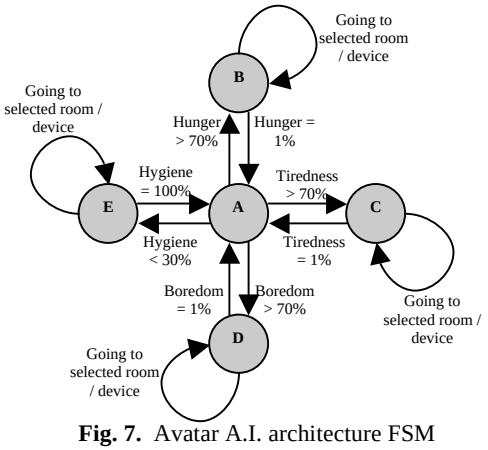


Fig. 7. Avatar A.I. architecture FSM

Table 8. 2D-PES Avatar AI states

State	Actor Action
A	All needs outside thresholds. No action required from the A.I. Actor continues using current device or following user instructions.
B	Hunger need > 70%. Actor moves to a randomly selected device which can be used to reduce hunger, (refrigerator, oven).
C	Tiredness need > 70%. Actor moves to a randomly selected device which can be used to reduce tiredness, (sofa, chair, bed).
D	Boredom need > 70%. Actor moves to a randomly selected device which can be used to reduce boredom, (phone, television).
E	Hygiene need < 30%. Actor moves to a randomly selected device which can be used to raise its hygiene level, (shower, washing machine).

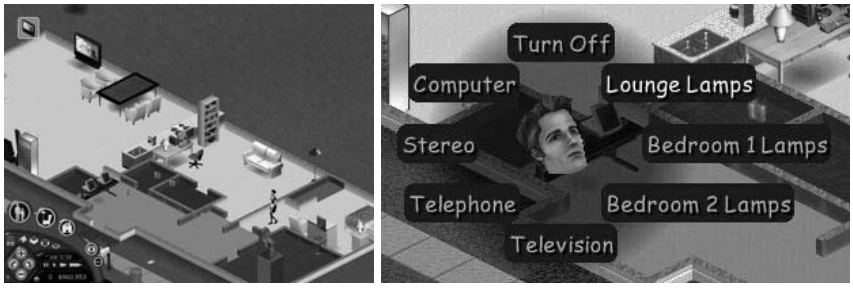
Simulation Environment. The simulated iDorm environment, (Fig. 6) contained an avatar inhabitant, (represented by a magenta circle), and numerous static, (grey rectangles) and pervasive objects, (orange/green rectangles). As the avatar interacted with an object, the device changed colour showing the state of an attached sensor, (interaction amounted to switching appliances on/off). If an object was green the attached sensor equalled 1 (high). If orange, the sensor equalled 0 (low). The avatar interacted with objects in response to the level of four 'needs' attributes, (hunger, tiredness, boredom, hygiene), which influenced activities performed (Fig. 7, Table 8).

2.2 Commercial Game-Based PES

Design Rationale. A three-dimensional PES, (3D-PES), created by modifying an off-the-shelf copy of the *Sims* computer game, (Maxis/EA Games, 2000). Apart from the 3D graphics and supporting tools, a particularly attractive feature in the *Sims* was the fairly realistic behaviour of inhabitants. Simulating the behaviour of people is a considerable challenge, the game avatar artificial intelligence, (AI) uses a "competing weighted behaviour" approach. One possible variation of this technique used for the avatar AI in the 2D-PES, is shown in Fig. 7 above. Using a series of weights to create an artificial bias in decision making, an artificial personality representative of a

person of any gender or age can be created for an avatar, and with additional research characteristics such as psychological/mental illness, could also be modelled. A PES could create a safety barrier, allowing agents/environments to be tested on a target audience, without placing real people at risk. Possibilities also include social research applications, (e.g. observing people with conflicting personalities living together).

Simulator Architecture. In more technical terms this system consisted of a five room environment, again modelled on the iDorm2. Each object and person contained in the environment was controlled by at least one thread, placed on a stack and run in sequence by the game. Object threads were used to regulate the animation displayed by the game virtual machine [6]. Most objects could only access their own threads, so for example a television couldn't access information contained in a thread for a lamp. To create a *Sims*-based PES, the original program code had to be modified so objects could access threads for other devices and information contained within. For this stage of the project, the most efficient way to achieve this was to program a single *Sims* object to act as a 'remote-control' for other pervasive devices. The 'Dumbold Voting Machine' [7] an add-on device available online, was modified to act as a remote-interface, and once re-programmed stored the current state of each pervasive object in the environment to memory. Agent code ran from the voting machine thread prompting state changes to other objects as required. Agents determined when to make changes using sensor values, updated on each thread cycle. Menus from the voting machine were re-programmed providing a manual interface to force actions performed by a *Sims* avatar, (Fig. 10).



Figs. 9-10. The *Sims*-Based 3D-PES & A Modified *Sims* Object Interface

Simulation Environment. The *Sims* game is a simulated domestic environment, designed to model one or several people living their daily lives. The original program allows a player to design, build and furnish a house to their own specifications, using numerous pre-programmed materials and objects available in the game libraries. Using a game to create a PES introduced several advanced features that add greater realism to the environment, but were too minor to allocate resources for programming into a bespoke system; including, avatars who randomly visit the virtual home.

3 PES Comparison

Creating two different simulators allowed a comparative study. The results intending to expose issues involved in modifying games, and relative advantages/disadvantages

compared to a bespoke system. We discussed the performance of the simulators under four criteria we regarded as important for pervasive computing applications;

- **Environment Representation.** This criterion covered how realistic the simulator graphics used, made the environment and its objects appear to be.
- **Avatar Representation.** How life-like were the movements and behaviours of the avatar inhabiting the virtual environment? How advanced was the AI used? Where actions purely reactive or could the avatar make decisions?
- **Agent Representation.** Perhaps the most important feature of a PES, allowing observation of environment changes, to evaluate how test-agents perform. How was influence of agents operating in the environment shown?
- **Programmability.** How easy was it to program? Did the platform used contain restrictions, which caused problems during simulator development?

3.1 Comparison Results Table

Table 11. The strengths and weaknesses for both simulators under the chosen criteria

Bespoke PES		Game-Based PES	
Environment Representation			
☑	The speed-of-time could be changed.	☑	Realistic 3D *Sims* graphics and animations.
☑	The simulator could be paused.	☑	The speed-of-time could be changed.
☑	Dawn and dusk are represented with natural-style changes to light levels.	☑	The simulator could be paused.
☒	Two-dimensional Java graphics made some features difficult to identify.	☑	Objects were added / changed in the environment quickly using the *Sims* game features.
☒	Added/changing objects in the environment required modifying code.	☑	Lights and windows both illuminated rooms naturally, with stronger illumination in areas closer to the light source.
☒	Lights, radiators and windows affected the environment evenly rather than having weaker influence in proportion to the distance from the source object.	☒	While dawn and dusk were represented with changes to light levels in the environment, the transitions occurred too quickly to appear natural and sometimes caused sensors to miss events.
Avatar Representation			
☑	Four A.I. 'needs' variables were used.	☑	Eight A.I 'needs' variables plus social values.
☑	Avatars interacted with objects by standing next to or sitting on them.	☑	Thought bubbles showed an avatar's desires.
☒	No visitors were available.	☑	Animations show avatar interaction with objects.
☒	Avatars didn't show any emotion.	☑	Avatars can visit and/or leave the environment.
☒	Inhabitants couldn't leave the environment.	☑	Avatars performed event chains, (e.g. to eat they walked to the fridge, cooked a meal at the stove, then ate at the table).
☒	Avatars didn't perform event chains for tasks, instead only using a single object.	☒	Avatars were messy, leaving rubbish around in unnatural locations and ignoring rotting food.
Agent Representation			
☑	Colour changes to objects made state changes easily visible.	☑	Animations make state changes easily visible.
☑	Agent activities were recorded to file.	☑	Multiple states were available for some objects.
☑	Agents were loaded from a 3rd party Java program file, with the server blocking any direct simulator contact.	☑	Agents could make modifications to the environment faster than for the bespoke system, since the code was part of the simulator program.
☒	Only on/off states were possible.	☑	Objects could randomly breakdown.
☒	Sensors were invisible in the simulator.	☑	A 'control' object in the *Sims*-PES allowed agents to be tested by forced avatar interaction.
☒	The time required for messages from the simulator and the agent program to be passed across the server created a short lag-time, before the environment	☒	SimAntics agent code had to be programmed directly into the simulator. Special sections were created during re-programming to facilitate this.
		☒	It was not possible to create output files.

	reacted to any required changes.	☒	Sensors couldn't be seen in the simulator.
Programmability			
☑	No restricted code.	☑	Changes to object code could be made quickly by re-structuring behaviour trees.
☑	Java is a multi-purpose OO language with many libraries and facilities.	☑	Edits occur in real-time with the game running.
☑	A popular programming language with programming guides and IDE tools.	☒	Little code other than *Sims* objects was editable.
☒	Since everything had to be written from scratch programming took a long time.	☒	SimAntics is a visual programming language designed for the *Sims* game, so is very specific in which operations could be performed.

3.2 Comparison

Environment Representation. The game-based simulator was created by modifying an existing infrastructure, adding and removing code-fragments where necessary. By using the *Sims* as a template the game-based PES objects looked realistic and could randomly breakdown where appropriate. Avatars had the appearance of real-people, interacting with the environment using realistic motions. The bespoke Java simulator visualized the same objects, but the devices were only shown as rectangles. This could cause confusion in complex demonstrations, if an observer had no knowledge of the layout in the iDorm environment. The 3D simulation of the *Sims*-PES also included features lost in the bespoke 2D-PES; For example objects mounted on walls.

Avatar Representation. Both simulators used 'needs' attributes to create an artificial intelligence for avatars inhabiting the environment. The bespoke system was simpler using just four variables to produce a reactive A.I, whilst the *Sims* game used eight 'needs' attributes, plus additional variables allowing some deliberation by avatars in social situations. Small thought-bubbles appear above the head of a *Sims* avatar showing that character's mood and/or desires. In a PES this feature can be used to observe how an agent/environment design affects the emotional state of avatars, possibly revealing areas requiring improvement.

Agent Representation. The *Sims* uses pre-programmed animations for many of its objects, to visually display any state changes, (e.g. switching a television on/off). Most of these animations were still present in the 3D-PES, although some were modified or removed for operational purposes. As explained earlier, the bespoke simulator also used animation on a much simpler scale, limited to two animations representing the on/off states of pervasive objects. By re-programming some of the original SimAntics code, some 3D-PES objects could visualise several new states and give different responses from normal to certain scenarios, allowing agents to remotely control devices. The 2D-PES system created output files recording the activities of the avatar and agents interacting with objects in the environment during testing. It was not possible to include this feature in the 3D-PES due to the restrictions and limitations of the *Sims* code. The simulator could use a feature of the *Sims* game, which saved/re-loaded the current environment, but no data was recorded to file.

Programmability. *Sims* object code was programmed using the visual language SimAntics, running from the Edith Virtual Machine [6] [16]. A new piece of code was added to the original game, where SimAntics agent programs could be attached directly to the simulator. Components were added to the bespoke 2D-PES ensuring

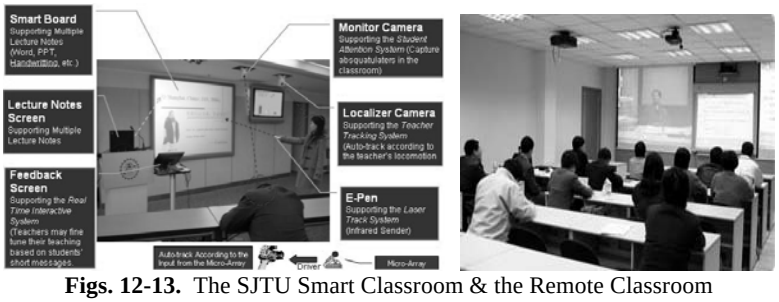
agent programs could be loaded into the simulator from an external Java file, written by a developer. The *Sims* game required re-programming before objects could be controlled by an agent. This wasn't a problem for the bespoke PES.

3.3 Comparison Summary

Criteria such as 'Computational Performance' were intentionally omitted from this evaluation. The aim was to accurately simulate a real-world environment; therefore at this stage of the research we were not concerned with simulator efficiency. Features such as ambient light changes representing dawn and dusk and the needs attributes for the avatar AI were programmed into the bespoke simulator, while the *Sims*-based simulator had many of these minor-features included as part of the original game. The *Sims* uses a more advanced AI system for avatars than the bespoke PES. A single inhabitant was used to test both PES environments, partly to reduce the programming requirements for the bespoke system, but mostly to allow each program to be tested using the same set of rules, allowing a more accurate comparison. Neither PES actually displayed sensors in their simulation. This was a decision made during the development stage to reduce the complexity of the simulated environments for this stage of the project. A next-generation PES will include sensors in the simulation, allowing several to be attached to a single object.

4 The Next Step

Our immediate aims are to extend the work to simulate the smart-classrooms (see Figs.12-13) used in the Open eLearning Platform in Shanghai, which supports more than 15000 learners. The platform delivers fully interactive lectures to PCs, laptops, PDA, IPTV and mobile phones from high-tech teaching rooms known as smart classrooms. The Essex iDorm includes a study-room, which is typical of the domestic space where remote learning from Shanghai can be delivered. Using the modified simulator, we aim to investigate how a virtual classroom might be best created to give teachers and distributed learners a sense of sharing the same space.



Figs. 12-13. The SJTU Smart Classroom & the Remote Classroom

With the simulator, and emotion sensing developed in another project [9], we will also investigate how learner's emotion evolves and affects the activities in the classroom [15]. This is part of a broader research strategy seeking to link a next-generation game-based PES with a real-world test-bed, creating a mixed-reality environment. Some advantages of this set-up would be allowing virtual avatars to

interact with real-world objects and vice-versa. Additionally, using virtual sensors to augment real-world counterparts should allow; a) more complex environments to be simulated; b) innovative devices and functions to be evaluated ahead of realisation, (particularly speculative devices that may not physically exist); c) the experience of playing games made more exciting by connecting it to the physical environment. We are also looking at how PES might be situated in globally distributed simulations such as Second Life [10] and Project Darkstar [17].

5. Conclusions

In this paper we described the first stage of an ongoing investigation into modelling pervasive environments with computer game technology. Primarily this stage focused on whether it was possible to take advantage of high quality graphics, and pseudo-realistic avatar behaviours, provided by computer games, to create a pervasive environment simulator, (PES). We discovered a PES had numerous advantages over a real-world test-bed, most notably; a) the speed-of-time was variable, allowing testing to be performed faster; b) event and sensor readings within the environment could be recorded, allowing experiments to be replayed in full or partially; c) environmental parameters in the virtual home could be specified, allowing agent programs to be tested under identical conditions; d) it has proven to be less expensive, in terms of cost, floor-space and maintenance requirements than using a real environment. We also noted that: a) several researchers could simultaneously use their own copy of a PES, customized to their experimental requirements; b) a PES would be easily portable and presentable, unlike a full-scale real-world environment; and c) a PES eliminates the problem for computer scientists who develop pervasive agents but lack skills required to build real-world devices to test the code.

To test our hypotheses we assessed whether modifying a computer game, had greater benefits than creating a bespoke PES or using a real-world environment to test pervasive agent programs. On comparison, (Table 11) for all but one of four criteria, a game-based PES would be more beneficial for pervasive computing research, than a bespoke counterpart. Our hypotheses were proven after modifying a retail-copy of the *Sims* computer game. A PES boasting high-level graphics and artificial intelligence was created in the same timeframe as a bespoke Java-based simulator, using simple two-dimensional visualisations and more basic A.I. Since the original game code was written by a team of developers with their own ideas, a single person could modify it into a PES but still maintain the ideals of the original programmers, preventing any personal bias from being introduced into the new simulator. In more scientific terms, the more natural behaviours of the *Sims* avatar and devices are a significant advantage to environments incorporating learning agents. Game A.I. can also be programmed with weights to let an avatar mimic a person of any age or gender. Such features are reflective of the real-world, so must be included in a simulated test-bed useful for developing pervasive agents and environments.

Finally, our immediate aims are to extend this simulation work to the creation of mixed reality distributed shared spaces, involving a richer combination of simulated people and appliance based agents. Initially, we plan to create a virtual classroom, based on our well proven Open eLearning and iDorm platforms. Our longer terms

aims are to extend this simulation so that it can be used, not only to develop intelligent environments but to study the symbiotic relationships that evolve in such spaces. We hope that this paper will have shown that game technology offers a means to provide an environment that has the potential to support such exciting research.

Acknowledgements

We are pleased to acknowledge the support of Electronics Arts who supplied the Edith editor for the *Sims*. Thanks are also due to Bernard Horan, Sun Microsystems, Michael Gardner, Chimera Socio-Technical Research Institute and Ruimin Shen, Shanghai Jiaotong University, who have greatly motivated this work by describing longer term visions, which we hope to address in future phases.

References

1. Callaghan V., Clark G., Colley M., Hagras H., Chin J.S.Y., Doctor F. *"Intelligent Inhabited Environments"*, BT Technology Journal, Vol.22, No.3 . Klywer Academic Publishers, Dordrecht, Netherlands, July 2004.
2. Callaghan V., Woods J., Fitz S., Dennis T., Hagras H., Colley M., Henning I., "The Essex iDorm: A Testbed for Exploring Intelligent Energy Usage Technologies in the Home", Proceeding of the 3rd International Conference on Intelligent Green and Energy Efficient Building & New Technologies, International Convention Centre, Beijing China, 26th-28th March 2007.
3. Chernett,P., Callaghan,V, Colley,M.J., Duffy,N.D., Edwards,I., Herd,J.T., Hunter,J., Lane,D.M., Penrose,J., Randall,G.W., Smith,D., Smith,J., Standeven,J., Whittaker,G.A., Wood,A., *'Mixing Simulated and Real Subsystems for Subsea Robot Developmen*t', IEEE International Conference Oceans 98, Nice, France, 1998.
4. Clarke G., Callaghan V., "Ubiquitous Computing Informatization, Urban Structures and Density", Built Environment Journal, Vol. 33, No. 2, 2007.
5. Cook, Diane J., Youngblood, Michael, Heierman III, Edwin O., Gopalratnam, Karthik., Rao, Sira., Litvin, Andrey., and Khawaja, Farhan. *"MavHome: An Agent-Based Smart Home."* In Pervasive computing: first international conference, Pervasive 2002. Ed. Friedemann Mattern, Mahmoud Naghshineh. Zurich, Switzerland, August 26-28, 2002. Berlin: Springer, 2002. p521 – 524.
6. Forbus, Kenneth D., Wright, Will. *"Some notes on programming objects in The Sims™."* Northwestern University, 31 May 2001.
7. Hopkins D., Dumbold Voting Machine, *http://www.donhopkins.com/drupal*, Retrieved: 19th June 06.
8. IIEG, iDorm2, *http://iieg.essex.ac.uk/idorm2/index.htm*, Retrieved: 18th March 2007.
9. Leon E., Clarke G., Callaghan V., Sepulveda F., "*A user-independent real-time emotion recognition system for software agents in domestic environments*", Engineering Applications of Artificial Intelligence, Volume 20, Issue 3, Pages 337-345, 2007.
10. Linden Lab., Second Life, *http://www.secondlife.com*, Retrieved: 5th April 2007.
11. Marinelli D., Pausch R., LaForce J., *"Entertainment Technology Center,"* IEEE Multimedia, Multimedia At Work, Eds. Catarci T., Little Thomas D. C., Vol. 7, Issue 4, p78 – 81, Oct–Dec 2000.
12. Microsoft Game Studios, *"Microsoft Flight Simulator 2004: A Century of Flight,"* CD.
13. Overmars M., *"Teaching Computer Science through Game Design,"* IEEE Computer, Volume 37, Issue 4, IEEE, pages 81 – 83, April 2004.

14. Ou Shichao, Karuppiah Deepak R., Fagg Andrew H., Riseman Edward, An Augmented Virtual Reality Interface for Monitoring of Smart Spaces, *Second Annual Conference on Pervasive Computing and Communications, Orlando, Florida, 14-17th March 2004*.
15. Shen L, Leon E, Callaghan V, Shen R "Exploratory Research on an Affective eLearning Model", International Workshop on Blended Learning 2007 (WBL 07) 15-17 August 2007, University of Edinburgh, Scotland.
16. Sims Zone, The, Interview: Patrick J. Barrett III, *http://www.thesimszone.co.uk/interviews/index.php?ID=1*, Retrieved: 15th March 2007.
17. Sun Microsystems, Project Darkstar, https://games-darkstar.dev.java.net, Retrieved 20th April 2007.
18. Voth, Danna. *"Gaming technology helps troops learn language."* IEEE Intelligent Systems. Vol. 19, Issue 5, Sept-Oct 2004. p4 – 6.