3D VIRTUAL INTERFACE FOR UBIQUITOUS INTELLIGENT ENVIRONMENTS

Kevin I-Kai Wang*, Ian Yen-Hung Chen*, Waleed H. Abdulla*, Zoran Salcic*, Burkhard C. Wünsche

*Department of Electrical and Computer Engineering
University of Auckland, Private Bag 92019, Auckland, New Zealand
{kevin.wang, i.chen, w.abdulla, z.salcic}@auckland.ac.nz
FAX. +64 9 373 7461

Department of Computer Science
University of Auckland, New Zealand
burkhard@cs.auckland.ac.nz

Keywords: Virtual Environment, Intelligent Environment, Multi-Agent System, Game Technology.

Abstract

In this paper, a 3D virtual interface and its corresponding physical testbed are introduced. The physical testbed named Distributed Embedded Intelligence Room (DEIR) realises a true ubiquitous intelligent environment by embedding sensors, actuators and computing devices and providing means of communication through various device networks. With the support of multi-agent control system, the 3D virtual interface is able to act both like a remote control interface or virtual simulator of the room. The 3D virtual interface has demonstrated the ability to perform real-time operations from remote locations and to facilitate research progress by allowing virtual simulations.

1 Introduction

Ubiquitous Intelligent Environments have recently become a popular research spot for many disciplines. Enormous time and expertises are required to construct a physical intelligent environment. The long process of constructing a physical environment very often delays the research process or even makes the project infeasible. This motivates the need of having virtual replications of physical environments to achieve cost-effective researches. 3D virtual simulations have been used in many applications such as robotics [5, 15], e-commerce [9] and intelligent environments [10, 11]. Technologies used to create computer games provide a quick and effective way to develop 3D virtual simulations and have become popular in various research areas. It can be commonly found in multi-robot simulators such as the USARsim project [15] which created virtual hazardous environments for the simulation of search and rescue robot operations. Faust et. al. [5] also used video game technology to perform Human-Robot Interaction (HRI) and collaboration studies. There are also many virtual simulators available that provide remote interfaces for teleoperating devices. An example is the real time 3D interface for remote control mission planning of unmanned aerial vehicle developed at the University of Auckland for the Defence Technology Agency (DTA) of New Zealand Defence Force [3].

In this paper, a novel 3D virtual environment is proposed and implemented using game technologies and multi-agent techniques. Game technologies have large user community and frequent update. They can easily provide interfaces with high quality graphics and network communication ability, which suit the objective of designing a high quality remote interface and virtual simulator. A 3D model of a physical ubiquitous intelligent environment named Distributed Embedded Intelligence Room (DEIR) has been constructed using 3dsMax [1]. The 3D model is exported and then rendered using an open source graphics engine, Ogre3D [7]. By integrating with the Multi-Agent control System (MAS) of DEIR, the 3D visualisation can be used not only as a remote control interface, but also as a virtual simulator for realistic experimentation.

The rest of the paper is presented as follows. Section 2 describes the physical intelligent environment and the control system architecture. Section 3 explains the development tools, communication protocol and system architecture of the 3D virtual environment. Section 4 presents the system evaluation and section 5 concludes the paper.

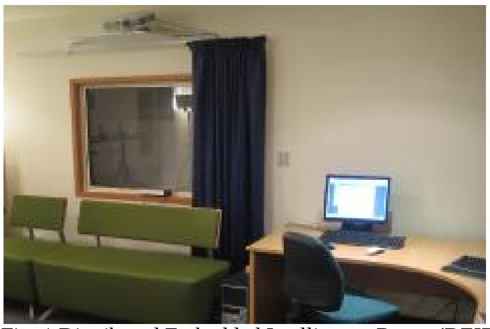


Fig. 1 Distributed Embedded Intelligence Room (DEIR).

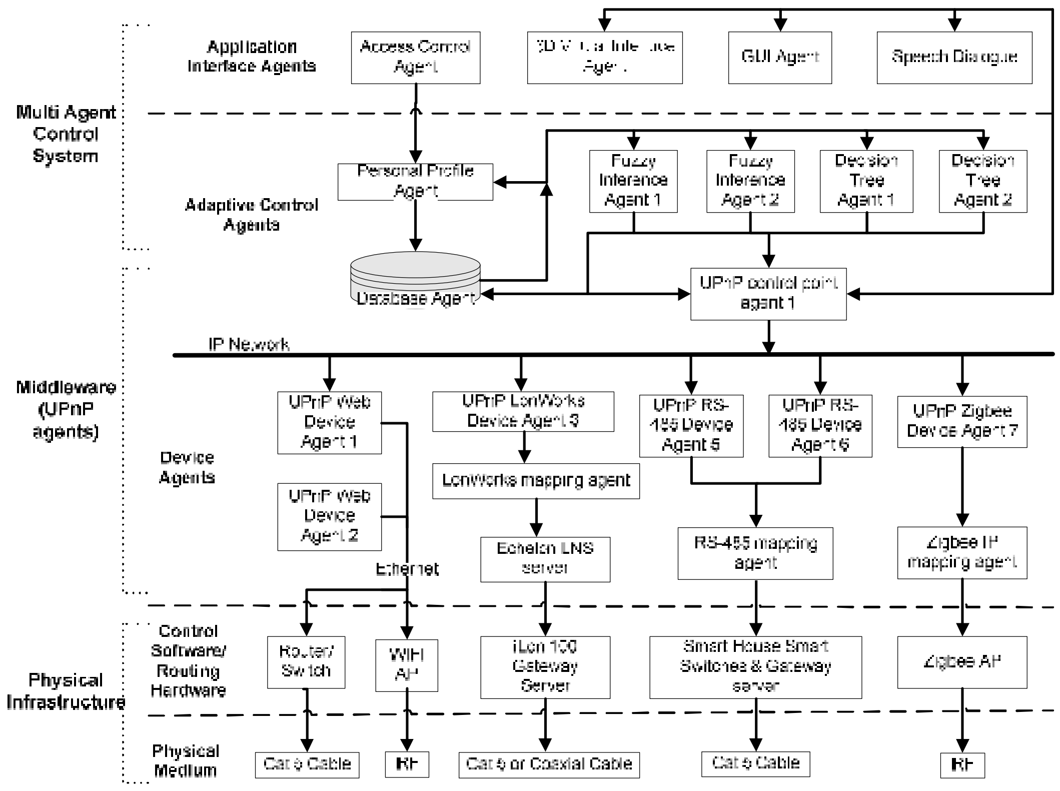


Fig. 2 DEIR system architecture.

2 Physical intelligent environment

In order to perform realistic real-time evaluation, a ubiquitous intelligent environment testbed named Distributed Embedded Intelligence Room (DEIR) has been constructed at the University of Auckland [12]. A picture of DEIR is depicted in Fig. 1. A number of sensors, actuators and computing devices are embedded within DEIR and interconnected through various physical networks. Embedded sensors such as light intensity, temperature, humidity, pressure, smoke and motion sensors are spread within the environment to monitor environmental context. Embedded actuators are used to automate windows, curtains, dimmable lights and other office appliances. Fig. 2 illustrates the system architecture of DEIR from physical layer up to the user application layer. In general, the complete system architecture can be described in 3 components: the physical infrastructure, the middleware and MAS, and intelligent control components. Details of each component are discussed in the following subsections.

2.1 Physical infrastructure

In the current DEIR system architecture, four different physical networks, RS-485, LonWorks network [4], Ethernet/Wi-Fi, and Zigbee have been used to control the embedded devices. RS-485 peer to peer (P2P) network links most of the actuators for windows, curtains and dimmable lights and a small portion of embedded sensors such as motion sensors. RS-485 devices are managed by a smart switch device that resembles traditional switch interface. The smart switch contains a M16C micro-controller that enables RS-485 communication and certain low level automation. LonWorks network connects part of the embedded sensors and the connected devices are managed by an iLon100 gateway server. Ethernet/Wi-Fi forms the backbone communication for all the embedded and mobile computing devices. Zigbee is employed as the communication protocol for some embedded sensors, such as pressure sensor, and commercial office appliances.

Hybrid device networks improve system flexibility in terms of incorporating different ranges of hardware devices. However, it adds a burden to the high level multi-agent control system to communicate with different devices. Furthermore, it is even more difficult to replace an existing network or to introduce new network technologies from a system architecture point of view. In order to integrate hybrid device networks and provide common portal to the high level controls, a middleware level is introduced. More detail on the middleware is given in the next section.

2.2 Middleware and MAS

To integrate hybrid device networks, Universal Plug and Play (UPnP) is used as the middleware [11]. In general, UPnP is an IP-based device discovery protocol which enables automatic device discovery, configuration and management [12, 13]. Hence, each individual device network is mapped onto the IP network and controlled as one integrated network. To enable UPnP communication, two UPnP components, UPnP software devices and UPnP control points, need to be implemented. The UPnP software device is responsible for communicating with the physical devices. It is the software replica of the underlying hardware. UPnP control point communicates with the software device and passes necessary information to the high level modelling and/or user interface components. The UPnP components implemented in the control system are developed by using CyberGarage UPnP Java API.

In DEIR, a well known agent platform, JADE (Java Agent DEvelopment Framework), has been employed as the underlying framework of the high level multi-agent control system [14]. Since both JADE and UPnP are provided in Java, middleware components can be easily integrated as low level agents in the control system. As shown in Fig. 2, UPnP software devices and control point are implemented as part of the MAS. In general, MAS of DEIR mainly consists of three types of agents: Utility agents, Modelling agents and Interface agents. Utility agents handle things such as access control, user personal profile management and user activities data deposition. Modelling agents involve two types of intelligent agent namely fuzzy inference agents and decision tree agents. The modelling agents are data-driven and will perform a series of learning steps to reach the final model of user behaviours. These agents provide the system the ability to model user activities and to provide automatic controls to the environment. The modelling techniques are explained in detail in [12, 13]. Interface agents communicate with user control interfaces such as the 3D virtual interface and GUI interface.

The proposed 3D virtual interface has two operation modes, namely the virtual simulator mode and the remote interface mode. Different communication paths are provided for different operations. The communication stack for our proposed 3D virtual interface is shown in Fig. 3. When used as a remote interface, the server application of the virtual interface receives control commands from the client and passes the commands through to the virtual interface agent and UPnP control point agent. The control point agent recognises that these are remote control commands and passes them down to the corresponding device agents for the commands to be executed physically. Acknowledgements on accomplishing the commands will be sent back to the control point agent and virtual interface agent to update the 3D virtual interface. While configured as a virtual simulator, the control point agent will forward the commands to the database agent and direct acknowledgements will be sent to the virtual interface agent to update the virtual simulator. Virtual simulator is used to test different types of user interface such as speech dialogue system in absence of a physical environment. Learning simulation is also possible by including intelligent control agents in the communication path.

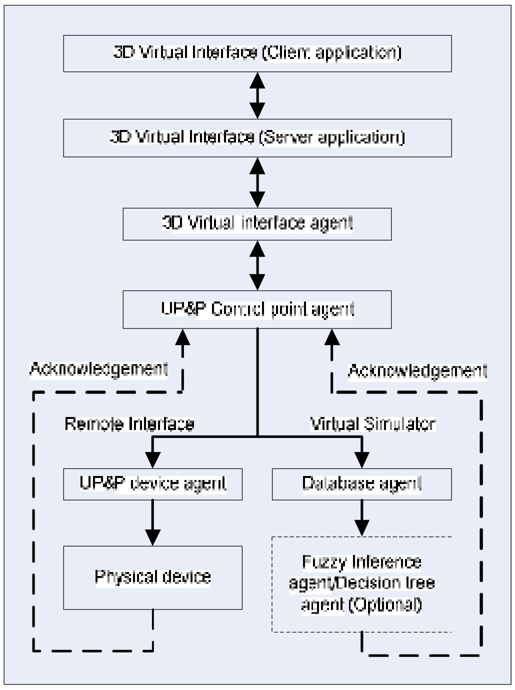

3D Virtual Interface (Client application)
3D Virtual Interface (Server application)
3D Virtual interface agent
UP&P Control point agent
Acknowledgement
Acknowledgement
Remote Interface
Virtual Simulator
UP&P device agent
Database agent
Physical device
Fuzzy Inference agent/Decision tree agent (Optional)

Fig. 3 Communication stacks for 3D virtual interface.

2.3 Intelligent control components

In the current multi-agent control system, two types of machine learning algorithms, Fuzzy Inference System (FIS) [12, 13] and decision tree, are incorporated as agents to model different types of user control behaviours. This is due to the fact that a single modelling technique may only be suitable for some control behaviours but not for all of them. The multi-agent architecture provides our control system the ability to apply different machine learning techniques to model different device control behaviours. The fuzzy inference and decision tree agents acquire the user activities data throu gh database agent and perform required data processing and learning steps. Once the learning steps are completed, both techniques will present the modelling results in terms of IF-THEN rules, which constitute a more coherent automatic control rule base. Further, traditional Rule-Based System (RBS) is employed in the UPnP software device agents to provide low level, hard-coded automation. Rules embedded in devices agents are low in priority compare to the rules from fuzzy inference agents and decision tree agents. This low level rule base provides a default set of automation before the high level learning is accomplished. It also ensures system response time in case high level agents crashed during runtime.

3 Virtual environment

A realistic, highly extensible virtual environment of DEIR is developed using game technologies and supports interactions from multiple remote clients. There are two main applications of this system: 1) The virtual simulator is designed to simulate the behaviour of DEIR in absence of the actual physical environment. 2) The virtual interface provides a 3D remote control interface which allows for teleoperating the devices installed in DEIR and presents a real time visualisation of the objects in DEIR. In addition, a controllable human character is introduced in the environment that offers end-users the option to view the simulation from different perspectives. It is also planned that the human character would support the simulation required in the study of occupant behaviour in the future.

3.1 System architecture

In order to support interactions from multiple remote users, a client/server architecture was implemented, as shown in Fig. 4. Network communication between the clients and the server is based on an ordinary Local Area Network (LAN) while the interactions between the server and DEIR MAS are performed through socket communication. In general, the server and clients will reside on different machines. The 3D virtual interface agent is more likely to be appeared in the same machine as the server to reduce the communication cost.

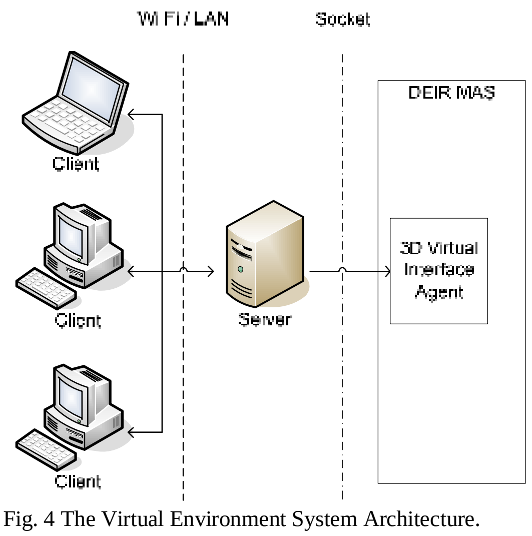


Fig. 4 The Virtual Environment System Architecture.

3.1.1 Server

The server manages connections to the remote clients in a Local Area Network (LAN) through Wi-Fi or Ethernet and handles all updates to the objects in the virtual environment. The server handles the communication between the virtual environment and DEIR by sending commands to the 3D virtual interface agent in DEIR MAS. DEIR MAS receives the commands and will in turn translate the commands into different network protocols, such as RS-485. It has been decided that the server is assigned a fixed IP address and port rather than utilising the Dynamic Host Configuration Protocol (DHCP), thus clients do not have to repeatedly search for the address of the server from time to time. The server is also a client itself, being able to visualise the virtual environment and allowing interactions for administration purposes. The system will stay running and accepts connections from multiple clients as long as the server is not terminated.

3.1.2 Client

The system architecture utilises a thick client model, reducing network traffic and relieving the computation load of physics calculation and scene rendering from the server. There are certain issues concerning this approach. The scene rendering performance becomes dependent on the hardware capability of the clients' machines, therefore having an influence on the portability of the system. Unlike desktop computers, other mobile devices such as PDAs do not have powerful Central Processing Units (CPU) nor are they installed with Graphics Processing Units (GPU). Although the game technologies we used are cross-platform on different operating systems, the problem lies on the amount of processing resources required to generate realistic graphics and physics on the client machines. However, achieving a high quality, real time simulation is a higher priority. Degrading the quality of the simulation using simplified graphical models or approximated physics calculations would inevitably produce inaccurate results, and increase inconsistencies between the simulation and the real world. By deploying the virtual simulation on the client machine, it allows scene renderings to be computed locally and realistic results are achieved without placing load on the network bandwidth.

The client can be started at any time. When started, it will continue to ping the specified IP address and port of the server until a response is received then joins the simulation. The statuses of the objects in the virtual environment on the client side will synchronise with the statuses of the objects on the server once the connection is established. A controllable player character will then spawn in the virtual environment and provides the end-user different views of the simulated room which will be discussed later. A client is able to leave the simulation at any time without disrupting the system.

3.1.3 3D virtual interface agent

The 3D virtual interface agent essentially provides an interface for the communication between the virtual environment and the DEIR MAS, encapsulating the underlying system structure and device control logics of DEIR. The agent is responsible for delegating the commands received from the virtual environment to the UPnP control point agent which in turn issues the commands to the correct hardware devices through the corresponding device agents. At the moment, the communication between the server and the 3D virtual interface agent is implemented using socket programming. A common control interface is considered as part of the future plan. More details are given in section 4.

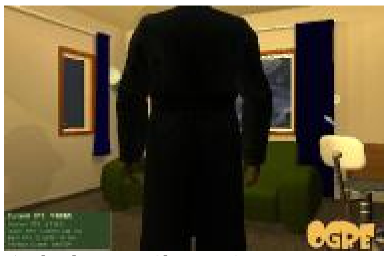
3.2 System technology and design

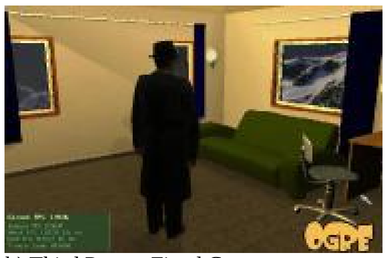
High fidelity graphics and realistic physics simulation have become the main challenges in developing realistic virtual simulations, requiring a significant amount of time, effort and expertise. We have developed the virtual environment of DEIR by adapting technologies used in the development of computer games. This has facilitated the creation of a high quality, multi-client, and interactive virtual simulation.

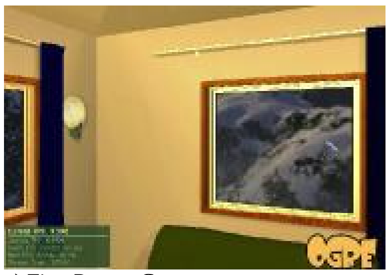
The open source Ogre3D Graphics Engine is chosen to be used as our 3D rendering engine. It has a large community support and various add-on projects that allow easy integration of physics and networking libraries necessary for this project. Ogre3D provides advanced features including material shader, scripted animations and other special effects [7]. Many existing libraries are available which support exporting 3D models from different 3D modelling tools such as Maya [1], 3dsMax [1], or Blender [2] to mesh files compatible with Ogre3D. The 3D objects in our virtual environment are created in real world scale using 3dsMax and mapped with photo textures taken from the objects inside DEIR. An additional scene file and material file will be generated along with the exported mesh files that describe the geometric locations and material properties of the 3D meshes. These are dynamically loaded at run time, providing an efficient way for extending the system to accommodate more graphical objects which saves time and effort. This implies that any arbitrary 3D model designers with no prior knowledge in the implementation of our system could also create and introduce new virtual objects without the need to write or to compile new code.

The open source openTNL [8] is used as our network library and sets up a Client and Server architecture for the system. It provides advanced ghost connections for establishing client-to-server connections, object replication, and also a simple, reliable and efficient Remote Procedure Call (RPC) framework for the communication between the client and the server. The main form of client/server communication is through openTNL˙s NetEvent messages. An event message is sent to the server from the client by using the RPC framework whenever the end-user performs an action that changes the state of an object in the virtual environment, for instance, opening a window. The server is responsible for broadcasting this event message to all the registered clients to ensure that all the statuses of the objects are consistent in every instance of the client simulation. The other form of communication is through openTNL˙s Server Object Replication. Objects on the server can be cloned to the clients. This is used to instantiate the human character object and to update its position and the orientation in the virtual environment when the end-users move it around using the keyboard.

Physics simulation is created using the Newton Game Dynamics Engine [6]. It provides real time simulation of physics environment. Newton Game Dynamics Engine enables users to create rigid bodies of objects in a world with gravitational force and modelling the objects with the appropriate mass, size, material and shape. When velocity, force, and torque are applied to an object, the dynamic behaviour of the object will be automatically simulated. Physics have mainly been used in our virtual environment to monitor collision detections and to simulate the behaviour of the furniture when exerted with force.


a) Third Person Chasing Camera.


b) Third Person Fixed Camera.


c) First Person Camera.

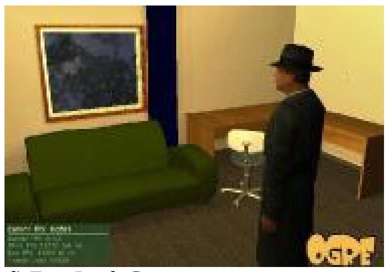
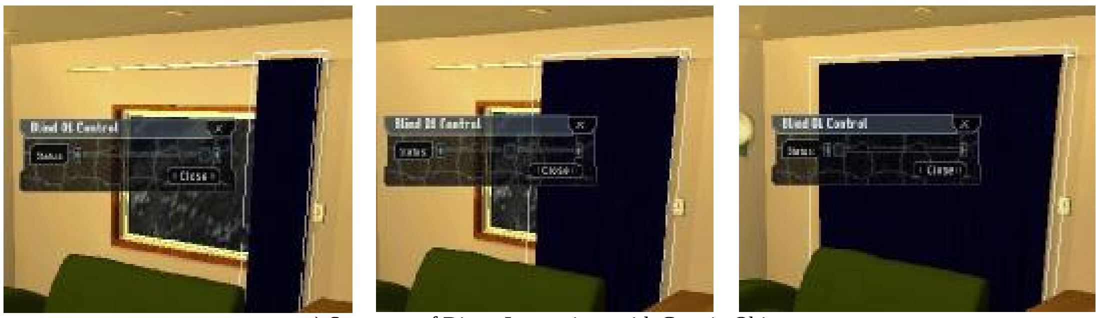
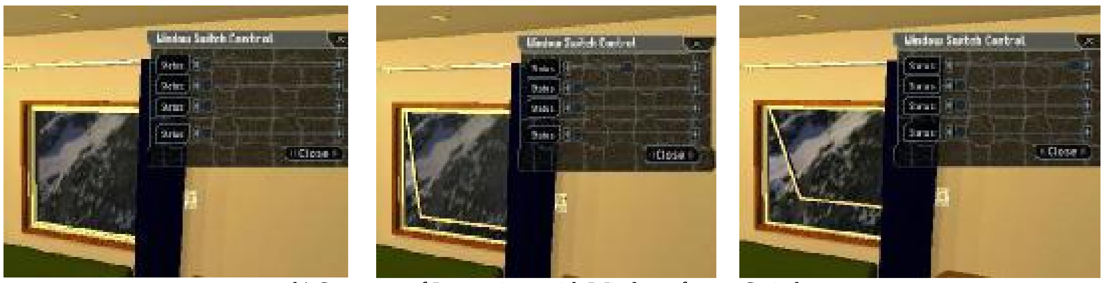

d) Free Look Camera.

Fig. 5 The four different camera modes of the system.



a) Sequence of Direct Interactions with Curtain Object.



b) Sequence of Interactions with Windows from a Switch.
Fig. 6 Dialogue Window for controlling objects in the simulated room.

3.3 DEIR simulation

In addition to providing a realistic graphical representation and physics simulation of DEIR, we took advantage of the interaction techniques supported from the use of game technologies. We present different camera modes for viewing the simulation, movement of the human character through standard keyboard inputs, and control of objects through graphical widgets.

3.3.1 Observing the simulation

The virtual environment enables the end-users to observe the behaviour of the simulated room from four different perspectives. This is established using the following four cameras modes: 1) Third Person Chasing Camera, 2) Third Person Fixed Camera, 3) First Person Camera and 4) Free Look Camera, as shown in Fig. 5. Upon joining the simulation, a human character is added to the simulated room. The human character is bound by the laws of physics and can be controlled using keyboard to move around the virtual environment. The view port will continuously be updated in the first three camera modes as the human character changes its position and orientation. These three modes permit interactions between the end-users and the simulated room, such as moving the human character or opening a window. Unlike the first three camera modes, the fourth camera mode of the system is an observer mode that allows the end-user to freely move the camera around the simulated room without any physics constraints. In addition, the end-user is not permitted to interact with the simulated room and therefore the statuses of the objects can not be altered by clients in this mode. The end-user will be able to switch from any camera mode to another at any time during the simulation by pressing the allocated keys on the keyboard.

3.3.2 Virtual remote control interface

Interactions between the end-users and the virtual environment are performed through a standard mouse. At present, controls have been implemented for lights, windows and curtains in the simulated room. These are flagged as 'Controllable Objects' in which their statuses can be manually altered by the end-users. Other types of objects include 'Selectable Objects' and 'Non-Selectable' objects. Selectable objects can be selected by users to provide a brief description of what they are and their current statuses while non-selectable objects are mainly background objects such as the walls and the ceiling. There are two ways to interact with the controllable objects in the virtual environment. The status of the controllable objects can be configured either by clicking on the object itself in the virtual environment as shown in Fig. 6(a), or by clicking on the corresponding switches on the walls as shown in Fig. 6(b). In both cases, a dialogue window will appear on the screen that allows the end-users to configure the status of a specific object or a group of objects connected to the switch, through graphical widgets such as push buttons and horizontal slide bars. After the end-user performs an action that alters the state of a controllable object, the object will be animated towards the specified new state using the same amount of time as it would when controlled in the real world. For instance, an action such as opening a window would slowly change the virtual window's orientation over time. Once the server is connected to DEIR through the 3D virtual interface agent, it can act as a virtual control interface of DEIR. All the actions performed by the clients will be reflected to the actual physical room and the virtual environment effectively presents a visualisation of DEIR for the remote end-users. Once the status of a controllable object in the simulated room is changed on the client, a command string is created and sent to the server.

Apart from the normal communication stack, the system currently also implements its own direct connection for communicating with the hardware which bypasses the DEIR MAS and the middleware layer. It is an additional communication path which can be used for hardware device testing purposes, for example, to identify whether an unexpected behaviour of a hardware device is the result of a bug from the software application or the hardware itself.

To resolve concurrency issues, at the moment only one command can be executed at any one time. The end-user is restricted to only change the status of a controllable object after the previous command has completely been executed. The dialogue windows are forced to become disabled in all instances of the simulation on the client sides to prohibit any actions during this time. It is important to note that this is only a simple concurrency management method designed at an application level to support current research needs. A more advanced control management system should be implemented in the future for managing commands issued by all applications to DEIR. This will be discussed further in section 4.

4 System evaluation

The objective of this project is to develop a 3D virtual interface that can be used for both virtual simulation and remote control interface. The system has been evaluated under remote interface mode for performance and real-time operation feasibility. The first evaluation investigates the performance of the 3D interface. Refer to Fig. 7, the 3D virtual interface performance is measured using Frame Per Second (FPS) with varying number of lights in a resolution size of 800x600 on three different machines with different hardware capabilities. A higher FPS value indicates smoother real-time 3D visualisation. Ordinary desktop computers, such as machine A and B in Fig. 7, have demonstrated a satisfactory performance of at least 6 FPS. In comparison, a laptop computer (refer to machine C) shows very poor performance of 1.5 FPS. To compromise this performance issue, fewer light sources can be used to achieve a higher frame rate at a cost of less realistic 3D model. The second evaluation shows the feasibility of real-time operation through Wi-Fi links. Remote client is located in a room that is 10 meters away from where the server resides, through one concrete wall. Device response time shows no difference as if the device is controlled through traditional switch interface and the virtual interface maintained synchronisation with the physical environment.

Hardware Specs\ No. of Active Lights	**15 lights (FPS)**	**9 lights (FPS)**	**4 lights (FPS)**	**2 lights (FPS)**
A	7.37	12.49	25.32	35.21
B	5.94	10.43	21.33	30.62
C	1.51	2.85	6.7	7.11

Key
A CPU: Intel Xeon CPU 3.73GHz, 1G RAM.
Graphics Card: Quadro FX3450/4000 SDI 256MB
B CPU: Pentium® 4 CPU 3.00GHz, 1G RAM.
Graphics Card: NVIDIA GeForce 6600 128MB
C CPU: Intel Pentium M 750 (Dothan), 1G RAM.
Graphics Card: ATI Mobility Radeon X600SE 128MB

Fig. 7 Performance comparison among different machines.

In order to use the 3D virtual interface as virtual simulator effectively, it is considered that another middleware layer (or a common control interface) should be introduced between the user interface application and DEIR MAS. This is due to the fact that high level applications can be implemented using many different programming languages, and having to write code with the same functionality in different languages is highly inefficient. Also, in the current virtual interface, the ability for concurrently controlling the room at the same time has been disabled. A more sophisticated control management mechanism involving command buffering and multi-threading should be employed and hence different devices can be controlled by different users at the same time. In addition to the support of Wi-Fi/LAN connection, remote control through internet connection is also considered as part of the future functionality. Further, more realistic human behaviour such as ‘sitting down’ and ‘running’ should be introduced to support more advanced simulations that model occupant behaviour. Finally, different interaction scenarios should be provided such as using the 3D virtual interface with touch screen devices.

5 Conclusions

In this paper, a novel ubiquitous intelligent environment, DEIR and its 3D virtual interface have been presented. DEIR consists of a number of embedded sensors, actuators and computing devices interconnected through four different physical networks, namely RS-485, LonWorks network, IP network and Zigbee network. High level multi-agent control system is employed to control the hybrid physical devices through a common middleware layer, implemented in Universal Plug and Play. The 3D model of DEIR is built using 3dsMax and realised through Ogre3D graphics engine. The design paradigm enables interface designer to design different virtual interfaces without knowing the underlying control architecture. The 3D virtual interface has demonstrated the ability to perform real-time operation from a remote location. It also facilitates the research process by allowing virtual simulations and experiments to be carried out in absence of a physical environment, with the support of multi-agent control system. To improve the flexibility and applicability of the 3D virtual interface, common control interface for high level applications and internet accessibility should be implemented.

References

[1] AutoDesk. Retrieved 18th February, 2007 from http://www.autodesk.com

[2] Blender.org Retrieved 18th February, 2007 from http://www.blender.org/

[3] B. Cervin, C. Mills, and B. C. Wünsche. A 3D Interface for an Unmanned Aerial Vehicle, *Proceedings of IVCNZ '04*, pp. 249-253, Akaroa, New Zealand, 21-23 November 2004.

[4] Echelon Corporation, "LonWorks Overview, " Retrieved 18th February, 2007 from http://www.echelon.com/solutions/overview/default.htm

[5] J., Faust, C., Simon, W. D., Smart, A Video Game-Based Mobile Robot Simulation Environment, *Proceedings of the 2006 IEEE*, Beijing.

[6] Newton Game Dynamics. Retrieved 18th February, 2007 from http://www.newtondynamics.com/index.html

[7] Ogre3D open source graphics engine. Retrieved 18th February, 2007 from http://www.ogre3d.org

[8] openTNL Torque Network Library. Retrieved 18th February, 2007 from http://www.opentnl.org

[9] C. T. Santos and F. S. Osorio, AdapTIVE: An Intelligent Virtual Environment and its Application in E-Commerce, in *Proceedings of the 28th Annual International Computer Software and Applications Conference*, pp. 468-473, 2004

[10] A. A. N., Shirehjini, A novel interaction metaphor for personal environment control: direct manipulation of physical environment based on 3D visualization, Computers & Graphics: Special Issue on Pervasive Computing and Ambient Intelligence, Vol. 28, pp. 667-675, 2004.

[11] A. A. N., Shirehjini, A Generic UPnP Architecture for Ambient Intelligence Meeting Rooms and a Control Point allowing for integrated 2D and 3D Interaction, in Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies, pp. 207-212, Grenoble, France, 2005.

[12] K. I., Wang, W. H., Abdulla, Z., Salcic, Distributed Embedded Intelligence Room with Multi-agent Cooperative Learning. *Ubiquitous Intelligence and Computing. Lecture Notes in Computer Science*, Vol. 4159, pp. 147-156, Springer-Verlag, Berlin Heidelberg, 2006.

[13] K. I., Wang, W. H., Abdulla, Z., Salcic, Multi-agent fuzzy inference control system for intelligent environments using JADE. in Proceedings of 2nd IET International Conference on Intelligent Environments, pp.285-294, 2006.

[14] K. I., Wang, W. H., Abdulla, Z., Salcic, A Multi-Agent System for Intelligent Environments using JADE. In Proceedings of the 1st IEE International Workshop on Intelligent Environment, pp86-91, 2005.

[15] J. Wang, M. Lewis, S. Hughes, M. Koes, and S. Carpin, Validating USARsim for use in HRI research, in *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, pp. 457 461, 2005.