

A Generic Framework to Exploit Virtual Worlds as Normative and Dynamic Interactive Spaces

D. Brota¹, I. Rodriguez¹, A. Puig¹ and M. Esteva²

¹Applied Mathematics Department, University of Barcelona, Spain

²Artificial Intelligence Research Institute, Barcelona, Spain

Abstract—*In this research we present a generic framework for behavior management in social virtual worlds. Due to the type of activities taking place in a social virtual world, it is important to rely on mechanisms ensuring that the virtual environment is prepared to be a dynamic space where participants are informed about activities evolution and where norms are used to organize participants' actions, to define actions' consequences and to prevent undesired participants behaviours. We define an interaction framework where model rendering and event capture are separated from decision mechanisms allowing to be used by an IA based module (e.g multiagent system) and exploited by different virtual world platforms. We present a prototype developed using Wonderland platform where we contribute with a new type of component named iObjectCell and a new scheme of multi-view model.*

Keywords: Virtual environments, Web3D and applications, information spaces

1. Introduction

Advances in internet technology get virtual world technology closer not only to home users (for entertainment purposes) but to wide-spread enterprises/institutions which have a powerful tool to develop remote activities between stakeholders like employees and clients in business, or teachers and students in a educational institution. Then, virtual environments have clear applications in e-governance, e-learning and e-commerce, just to name a few.

In this research, we work with social virtual worlds populated by participants that collaborate in order to achieve a common and individual goals [1]. Due to the type of activities taking place in a social VW, it is important to rely on mechanisms ensuring that the virtual environment is prepared to be a dynamic space where 1) participants are informed about activities evolution and where 2) norms are used to organize participants' actions, to define actions' consequences and to prevent undesired participants behaviours. We rely on virtual objects, populating the virtual world, to enforce norms and give assistance to participants.

Continuing with our line of research in the so-named intelligent objects [2] [3], we present a general framework of object behaviour control tied with an IA based external module and prepared to be exploited by several VW platforms.

This is done creating a specific module to capture participant interactions on objects populating the virtual world and connecting this module with an external and generic one in charge of deciding what should be the virtual object action. Decision depends on an organization-based multiagent system (MAS) which establishes the valid interactions participants may have and the consequences of those interactions [4][5]. In this paper, we present contributions that can be classified in two different groups. In the first one, we arrange general issues enumerated as follows:

- We ensure the compliance of norms by participants using virtual objects with an external module named iObject manager which decides whether participants comply established norms.
- We present a new object behaviour control scheme applicable to different VW platforms.
- We see VW as dynamic information spaces which exploit the virtual nature of the spaces, and the objects populating these spaces, allowing to represent things impossible to their real world counterparts.

The second group of contributions are related with the VW platform, Wonderland (WL)[6], chosen for the prototype:

- We have developed a multi-view scheme for objects in WL in order to allow that different users have different views of the same object.
- We incorporate a new type of Cell named iObject and in particular we incorporate the concept of Door in WL (iObjectDoor), until now inexistent in this virtual world platform.

This paper is organized as follows. Section 2 reviews the related work on norms management in virtual worlds, VW as singular information spaces and interaction schemes for dynamic virtual environments. Section 3 gives a general overview of the interaction and information space framework. In section 4, the proposed framework is evaluated using our own extension of Wonderland virtual world. Section 5 gives conclusions and future work.

2. Related work

Most of well-known virtual communities -such as Second Life, Active Worlds, Entropia and others- require participants to agree to the company' s terms of service in the signing up process [7]. Participants should understand the terms

and conditions to which they are agreeing as a member of that community. Most people don't read or are otherwise immune due to the lack of consequences. There are some types of incorrect behaviours that we think can be addressed programatically, that is contemplated in the design of the VW platform and ensured at deployment time. We propose to use intelligent objects as element helping users to comply norms and if it is necessary to prevent forbidden actions. For example, to block entry to people who is less than 18 years old in a special virtual room. WonderDAC is an extension module developed for Wonderland that allows to show or hide parts of a VW depending on the user and group profile [8]. In contrast to WonderDAC, developed to control discretionary access basing on users and group permissions, our approach is more general allowing, for example, the control of access to spaces based on the historic of user activities. For example, a norm establishes that a participant can not enter to the projection room unless he has bought a ticket for that room and session.

Part of our inspiration for a general interaction approach for objects populating a virtual world comes from the smart objects proposal [9] and the posterior work done by Jorinssen [10] [11]. Nevertheless, our approach is different to those because they worked with their own virtual environments named ACE (Agent Common Environment) and ALVIC (Architecture for Large-Scale Virtual Interactive communities), respectively. In this way, their object interaction approach is general in the sense it is independent of the final application but can not get out of their concrete virtual platform (they have their own scripting language and engine). Our interaction framework has been designed to be applied to different VW platforms such as Wonderland and Second Life. In this way, rendering and event capture continue being controlled in the concrete VW platform but the behaviour decision is managed in an external and generic manager.

Virtual worlds can be seen as singular information spaces where the virtual nature of the own space (e.g. floor) and the furniture (e.g panel) can be exploited in a special manner not possible for their real counterparts. For example, in the real world it is not possible to dynamically change tiles color in a floor to represent an agree/disagree position of participants in a discussion. This has been done in a recent work [12]. We aim to incorporate an added value to virtual objects allowing to give valuable information to participants. As an example in section 4.4, a door is visualized either green or red depending on the user trying to pass through. Accessibility issues can also be addressed in these information spaces, for example a noticeboard object adapts letter size depending either on user profile or on the distance between the user and the panel. Exploiting these native properties of virtual objects, we create rich and expressive social spaces.

We extend the dynamic conception of current VW platforms in which users are free to dynamically change aspects

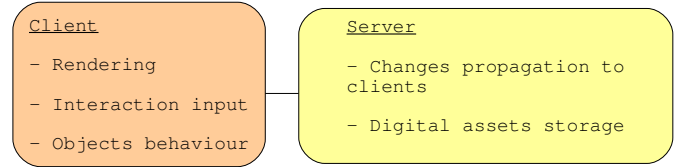


Fig. 1: Client and server side functionalities in a conventional VW platform

of the virtual world by means of built-in tools and scripting behaviours [13] [6]. Part of the unexplored promise of virtual spaces is their potential for automatic architectural flexibility. Our proposal is to extend the ability of a VW to dynamically change itself and exploit the virtualness of the space supporting the presentation of information, which would be impossible to do in the real world, and so provide a better support to participants on their activities.

3. Generic behaviour management

This section presents our framework for generic behaviour management and compare it with the approach of conventional virtual worlds. Our framework decouples event provider from event dealer (i.e behaviour handling) allowing a better support for normative and dynamic virtual worlds.

Figure 1 depicts functionalities included in a conventional VW platform based on a client-server technology. Clients take charge of rendering, interaction and behaviour handling (i.e. event capture and treatment). On the server side, digital assets are stored (in proprietary or standard format), and the server propagates client changes to the rest of connected users. A drawback of this architecture is that an object behaviour has to be reprogrammed when VW platform changes.

Our approach, presented in Figure 2, gets behaviour handling out of the VW platform. It is treated in an external module named iObjects manager. An iObject is a 3D entity populating the virtual environment which is exploited in two ways: it allows normal interaction as it would do in the real world (e.g approach/touch a door to open) and its virtual nature gives an added value to the provided information (e.g adapts dinamically color or size). More information on iObjects can be found in [2] [3]. In the virtual world, iObjects ensure participants norm compliance and give the user assistance during his participation.

iObjects' manager is designed to be used by several virtual world platforms. To do that, it is needed to develop an extension module, iObjects extension in Figure 2, in the VW platform that will communicate with the generic manager using a socket connection. Next section describes the extension module we have developed in Wonderland from Sun Microsystem and presents some simulation results.

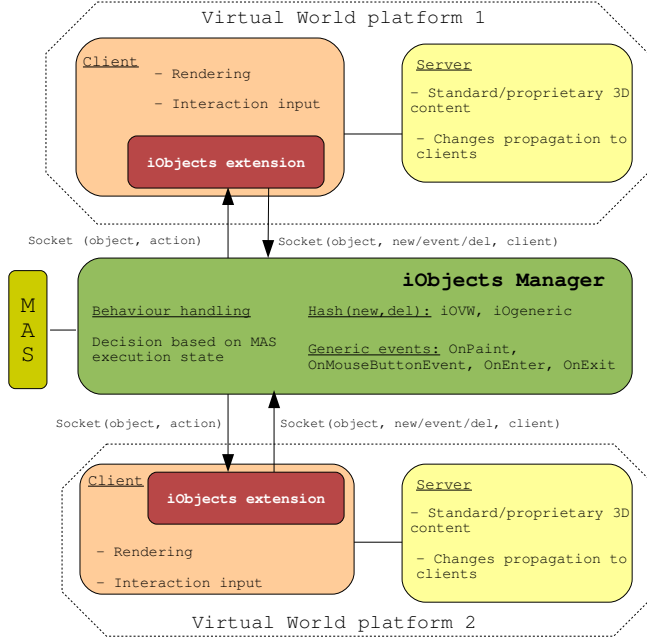


Fig. 2: Our proposal for decoupling behaviour management in VW platforms

As can be appreciated in Figure 2, an interaction with an iObject is captured in the virtual world client and it is sent, using a socket message, to the iObjects manager. The message indicates client identifier, object and event used to interact with the object. The iObject manager decides which iObject action (e.g. change color, size, trigger animation) has to be sent back to the VW. This decision is based on a response given by an organization-based multiagent system which establishes norms and possible interactions. The manager maintains a hash with iObject identifier in the concrete VW (iOVW) and its generic counterpart (iOgeneric). Currently, generic iObject events contemplated are OnPaint, OnMouseButton, OnEnter (an avatar enters in an area near to object's position) and OnExit (an avatar leaves an area near to object's position). Note that it is needed to do a mapping between concrete VW events and generic ones contemplated by the iObjects manager.

4. Prototype for evaluation

There are several VW platforms to develop an interactive virtual environment, Second Life, Active Worlds and Wonderland, to name a few. All of them consist of similar components such as avatars, buildings, scripting components and built-in features. We chose Wonderland because it was conceived to work with 3D standards, it is open-source and multi-platform (java-based). We have developed our prototype in WL 0.4 where 3D content is represented in X3D standard format. WL 0.5 works with COLLADA, a

well-extended 3D interchange format. We are now migrating to version 0.5 available only for developers.

Once selected the VW platform, we studied 1) how to incorporate iObjects in WL and 2) how to capture an event (i.e. an object interaction) in WL and communicate it to the external and generic iObject manager. In particular, our prototype presents results obtained using an iObjectDoor. In WL, doors are merely holes allowing to pass through them to avatars and so change from one room to another one. An iObjectDoor adds an additional nuance letting pass through it only avatars having permission, that is, avatars who comply with the norms established by the multiagent system shown in Figure 2. Our extension of WL platform requires to develop a new type of Cell named iObjectCell. Cells are the small unity of visualization in WL. A Cell represents a 3D volume and its properties. There are static and movable cells. For example, an avatar is a movable cell and furniture, walls and floor are static cells. As shown in Figure 3, cells are nested into a tree structure (i.e. scene graph). For instance, in a world you might have a WorldRootCell that represents the whole world, a room cell that represents a room, then cells within that room representing avatars, a phone and a door.

Every cell is registered on the Server, and if the avatar, controlled by the user, is within a cell vicinity, the model is sent to the client who comes to see it. In this way, models are only loaded when it is needed with the consequent saving on memory and time.

The server-side cell-representation is loaded at the server boot-trapping time. The server maintains the shared state of the world across participants (see scene graph in the left part of Figure 3). Clients are responsible for local renderings, using local files and by drawing objects with JAVA 3D APIs. Local changes in the scene can be performed without notification to the server side and so, to other clients. If local changes are notified, the server broadcasts a message to all clients describing the "kind of change" done. All clients see the same scene (e.g. table in Figure 3) under different points of view, depending on their avatar positions. In section 4.2 we present an extension of WL to allow multiple views of the same cell by different clients.

4.1 Extending WL with iObjectCell

In WL, each cell has four main components: a CellGLO (Game Logic Objects) on the server side, a Cell in the Client side, an intermediate Cell for client-server communication and a visual representation (i.e. 3-D model). Each CellGLO class on the server has a corresponding Cell class on the client which is responsible for rendering the cell and for capturing cell events. Each cell has been modelled in Blender, exported as X3D and converted to .j3s format, that is the final format loaded in WL 0.4 version. We have extended WL creating a new type of Cell named iObjectCell.

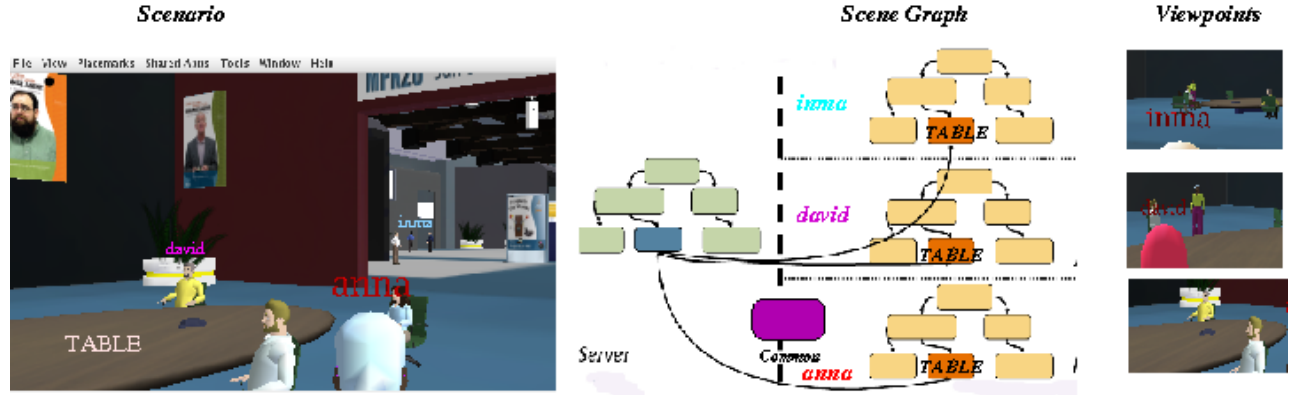


Fig. 3: Cell trees

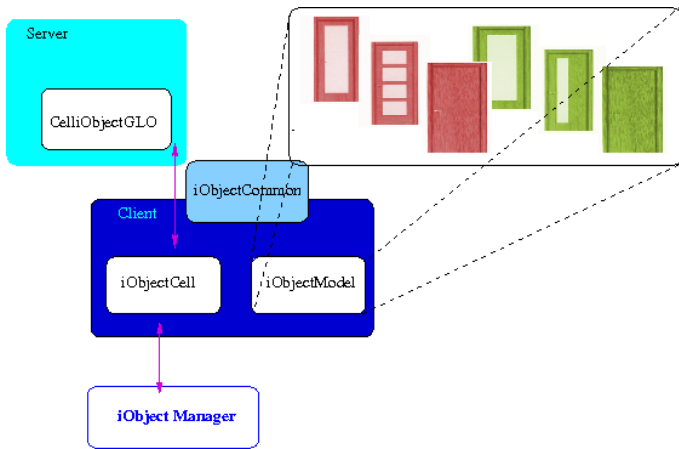


Fig. 4: Components of an iObject Cell

Figure 4 shows iObjectCell components: iObjectCellGLO, iObjectCell, iObjectCommon and the 3D model.

iObjectCellGLO subclass extends WL class StationaryCell, it communicates to the client-side the cell visual content and other setup data implementing the CellGLO.getSetupData() method. In addition, iObjectCellGLO controls an iObject different states and actions. Moreover, a set of opened clients' sessions is used to broadcast an specific action to everyone. For instance, if a door is opened by a client, the rest of the clients should see the door opening animation, although they are not allowed to pass through it.

iObjectCell subclass is an specialization of the client-side Cellclass. It implements the ExtendedClientChannelListener in order to handle cell events (e.g mouse, entry/exit vell). Both the visual representation and animation of the cell are managed by this subclass. For example, 3D tranformations of the cell geometric model for animation purposes.

iObjectCommon maintains both the current state of iOb-jects and the set of messages exchanged between the server

and the client side. It is also responsible for reading and parsing the XML cell file within the Wonderland File System (WFS). WFS is a set of XML files, structured in a similar way as unix file system, shared by the server and the clients that describes the layout and state of a world on disk. In these XML we store all visual models and material properties of the iObject, such as color and textures.

4.2 Extending WL with a multi-view scheme

Virtual worlds can be exploited as dynamic information spaces, for example, adapting the visualization of a virtual object depending on the participant profile or previous activities. We propose a cell multi-view scheme by keeping different 3D models of the cell. All clients share an indexed set of visual representations of a cell (red door, green door, glazing door, etc.), but only one is active for each client in a given moment. Figure 4 illustrates the set of models and visual properties of a door.

When an iObjectCell changes its appearance for a client, the client-side of the cell notifies the produced change to the server-side iObjectCellGLO subclass. The server should be informed in order to maintain the current iObject' visual appearances for all of the clients, due to the memory optimization on the client side is managed by the server. In this optimization, only the cells which are currently visible are loaded by the client and those which are no longer visible are deleted. The visible cells are determined by the avatar position. When the avatar walks around the world, previously non-visible iObjects can become visible now. In this point, the visual aspect of the iObject should be coherent to its last change.

4.3 A Simple iObject: iObjectDoor

The visual representation of a door has been modeled using Blender. An iObjectDoor is an iObjectCell that consists of a door frame, a door shape and a transparent bounding cylinder (See Figure 5). Entry and exit events on the iObjectCell are used to either enable or disable

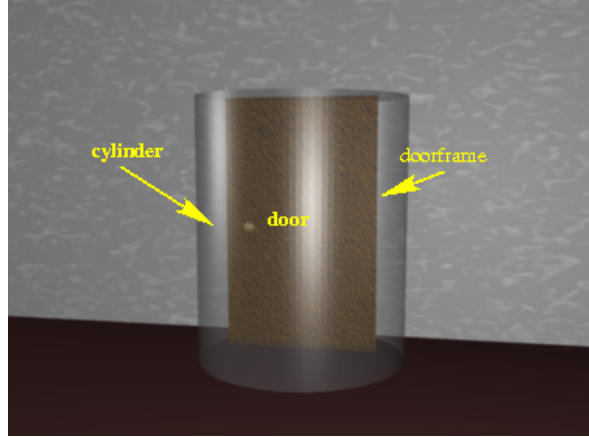


Fig. 5: Modeling an iObjectDoor visual representation

collisions between the avatar and the iObjectDoor. We use the bounding cylinder to forbid, if necessary, clients to pass through the door. In this way, the bounding cylinder actuates like an invisible wall.

Wonderland server handles collisions only on objects represented as triangulated meshes. Thus, iObjectDoor's bounding cylinder is represented as a mesh. Collisions are disabled or enabled whenever the WL client captures the enter or exit cell events, respectively. An enter cell event is triggered when an avatar gets into an established bound radius of a cell. On the contrary, an exit cell event is triggered when an avatar gets out bounded limits.

By default, the collision control of the iObject's cylinder component is enabled for all clients. Only on iObjectDoor entry events, the collision control is disabled when a client with access permission tries to pass through the door. Then, a door 3D animation starts to open the door. To avoid concurrent access to the door, it is not allowed any client to access to the threshold of the door meanwhile it is opening and closing. Two avatars should not go through the iObjectDoor simultaneously, even if both of them are allowed to do it. When the avatar has passed through the door, an exit cell event is triggered, a closing animation starts and the collision control of the door is enabled again. In addition, all avatars without access permission have always the collision control enabled so that they can never get closer to the door.

An iObjectDoor animation on the client-side is managed by means of 3D matrix transformations. In WL 0.4, another possibility is to import a .rtg file containing both the 3D model and the animation. When an avatar is near to the door and clicks the mouse over the door, the iObjectDoor captures the event and asks the iObjects manager whether the client complies norms allowing to access the room (e.g. in an auction, the buyer has paid registration fee). Then, if the answer is affirmative, the iObjectDoor runs the local animation and notifies it to the server so that the rest of

clients also visualize it.

4.4 Simulations

In this section, we show two simulations exploiting norm compliance and multi-view scheme for an iObjectDoor in Wonderland. Table 1 shows snapshots taken from two clients in successive instants of time (T1-T5). Client 1 sees the door in green because he complies with the norm allowing to enter the next room. Client 2 sees it in red because he does not comply the norm. Recall that iObject door is in charge of ensure norm compliance by means of iObjects manager connection to the MAS (multiagent system, see Figure 2).

Table 2 shows snapshots taken from three clients in successive instants of time (T1-T2). Client 1 sees a red door because he doesn't comply norms allowing to both enter the next room and see through the door. Client 2 sees a glazing red door because he doesn't comply norm to enter and complies norm to see through the door. Client 3 sees a glazing door in green color because he complies both norms.

5. Conclusions and future work

In this research we present a generic behavior management for objects populating a virtual world. We get behaviour handling out of the VW platform so that it is performed in an external module named iObjects manager allowing to be exploited by different virtual world platforms. An iObject is a 3D entity populating the virtual environment which is exploited in two ways: it allows normal interaction as it would do in the real world (e.g approach/touch a door to open) and its virtual nature gives an added value to the provided information (e.g adapts dynamically color or size depending on the client). Extending Wonderland platform, we have contributed with a new type of cell name iObjectCell and, in particular, we have developed an iObjectDoor which 1) allows to pass through it only to avatars complying norms established by a multiagent system and 2) offers a different visualization of objects (multi-view scheme) depending on the client.

As future work we plan to extend the iObject module with new types of intelligent objects (e.g notice-board, brochure) and test its functionality in other VW platforms such as SL or Active Worlds. In the current prototype iObjects are stored in the structure of directories managed by Wonderland server (WFS). For virtual world platforms which work with 3D standard content, we plan to offer a database of iObjects prepared to be incorporated to the VW at run-time.

Acknowledgements Partially funded by projects IEA (TIN2006-15662-C02-01), AT (CONSOLIDER CSD2007-0022), EU-FEDER, CICYT TIN2008-02903 and by the Generalitat de Catalunya under the grant 2005-SGR-00093. M.Esteva enjoys a Ramon y Cajal contract from the Spanish Government.

Table 1: iObjectDoor ensuring norm (pass through) and multi-view (color) scheme

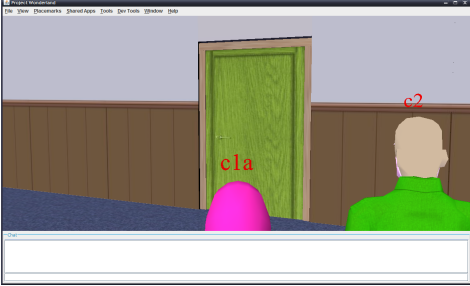
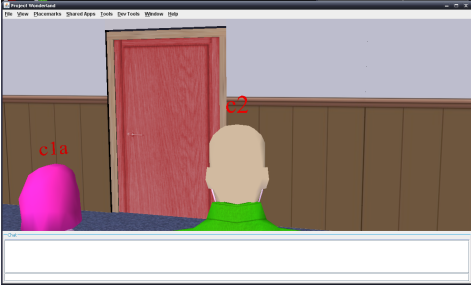
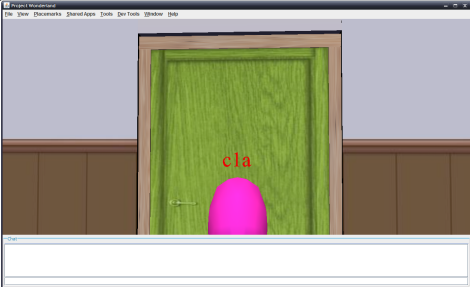
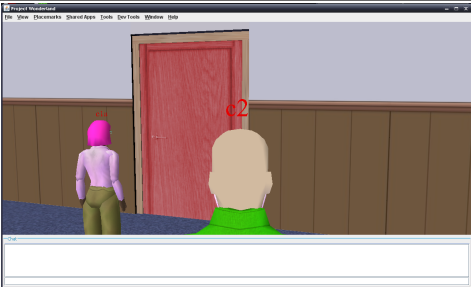
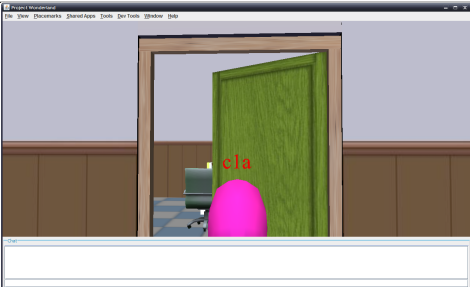
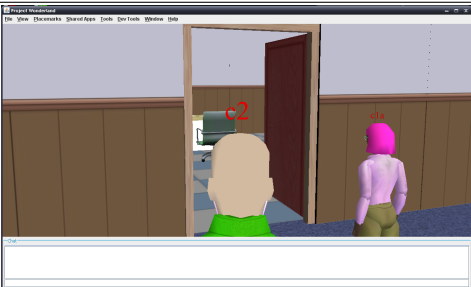
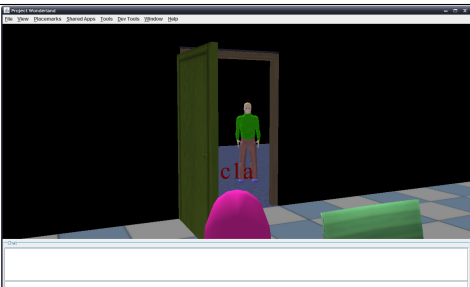
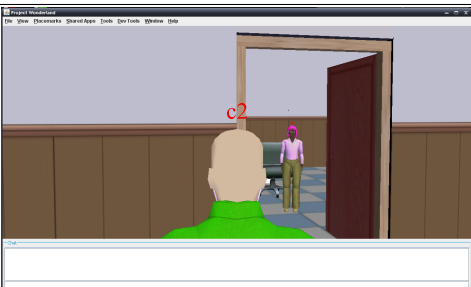
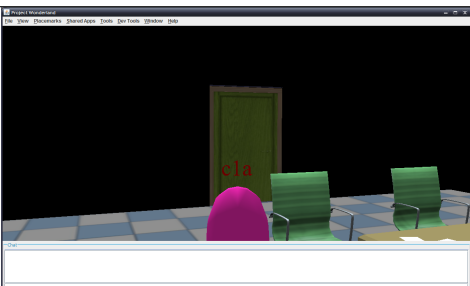
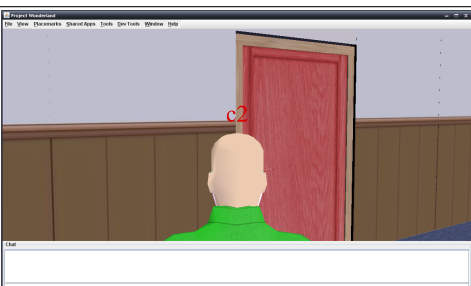
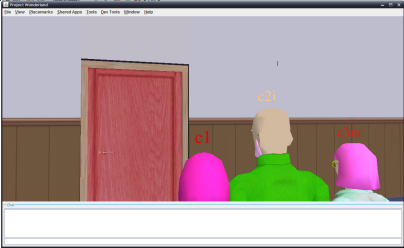
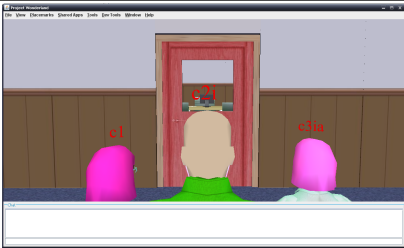
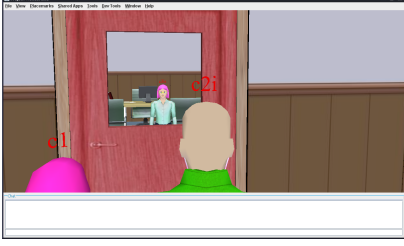
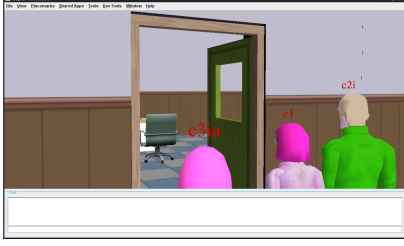
	Client1 complies norm to pass through	Client2 doesn't complies norm
T1		
T2		
T3		
T4		
T5		

Table 2: iObjectDoor ensuring norms (pass and see through) and multi-view (different color and type of door) scheme

	Client 1 (access denied) (visibility denied)	Client 2 (access denied) (free visibility)	Client 3 (free access) (free visibility)
T1			
T2			

References

- [1] R. Bartle, *Designing Virtual Worlds*. New Riders, 2003.
- [2] I. Rodriguez, M. Salamo, M. Lopez, J. Cerquides, A. Puig, and C. Sierra, "Completing the virtual analogy of real institutions via iobjects," in *CCIA*, 2007.
- [3] I. Rodriguez, A. Puig, M. Esteva, C. Sierra, A. Bogdanovych, and S. Simoff, "Intelligent objects to facilitate human participation in virtual institutions," in *Conference on Web Intelligence*, 2008.
- [4] M. Esteva, *Electronic Institutions: from specification to development*. PhD (UPC), 2003, ser. IIIA Monograph Series. IIIA, 2003, no. 19.
- [5] A. Bogdanovych, M. Esteva, S. Simoff, C. Sierra, and H. Berger, "A methodology for developing multiagent systems as 3d electronic institutions," in *Agent-Oriented Software Engineering VIII*, vol. 4951, 2008, pp. 103–117.
- [6] "Wonderland from sun microsystems," <https://lg3d-wonderland.dev.java.net/>, 2008.
- [7] "Second life term of services," <http://secondlife.com/corporate/tos.php>, 2008.
- [8] T. E. Wright and G. Madey, "Wonderdac: An implementation of discretionary access controls within the project wonderland," in *Tech report. Univ. of Notre Dam*, 2008.
- [9] M. Kallmann, J. Monzani, A. Caicedo, and D. Thalmann, "A common environment for simulating virtual human agents in real time," in *Proc. Workshop on Achieving Human-Like Behavior in Interactive Animated Agents*, 2000.
- [10] P. Jorissen, M. Wijnants, and W. Lamotte, "Using collaborative interactive objects and animation to enable dynamic interactions in collaborative virtual environments," *Conference on Computer animation and Social Agents*, 2004.
- [11] P. Jorissen, M. Wijnants, and W. Lamotte, "Dynamic interactions in physically realistic collaborative virtual environments," *IEEE transaction on visualization and computer graphics*, vol. 11, no. 6, pp. 649–659, 2005.
- [12] D. Harry and J. Donath, "Information spaces: Building meeting rooms in virtual environments," in *Conference on Human Factors in Computing Systems*, 2008, pp. 3741–3746.
- [13] D. Friedman, A. Steed, and M. Slater, "Spatial social behavior in second life," *Lecture Notes in Computer Science, Springer*, vol. 4722, pp. 252–263, 2007.