Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 651–658

ISBN 978-83-60810-22-4

Visualizing Agent-Based Simulation Dynamics in a CAVE - Issues and Architectures

Athanasia Louloudi
Modelling and Simulation Research Center
Örebro University, Sweden
Email: athanasia.louloudi@oru.se

Franziska Klügl
Modelling and Simulation Research Center
Örebro University, Sweden
Email: franziska.klugl@oru.se

***Abstract*—Displaying an agent-based simulation on an immersive virtual environment called CAVE (Cave Automatic Virtual Environment), a human expert is enabled to evaluate the simulation's dynamics from the same point of view as in real life - from a within perspective instead of a birds eye view. As this form of face validation is useful for many multiagent simulations, it should be possible to setup such a system with as little effort as possible. In this context, we systematically analyse the critical issues that a realization of such a system raises. Addressing these problems, we finally discuss design aspects of basic framework architectures.**

I. Introduction

PERFORMANCE evaluation is important for agent-based simulations [1]. In order to ensure that the model used is able to produce reliable and plausible results, significant oversight from the human expert is required. However, this could easily become a source of great expense, especially in cases involving agent behaviour in explicit metric space such as in pedestrian simulations. To reduce this cost, an immersive visualization, based on multiple interlinked views with different levels of detail, could be considered as a useful tool for evaluating the plausibility of simulation models at the agent level. Due to the high degree of immersion, face validation [2] can be very much facilitated. In this form of validation, one or more human experts assess the model based on animations, simulation output or from a within perspective (e.g., agent's view). Debugging and model evaluation is then clearly benefited by zooming into the system, even into the agents for observing the ongoing dynamics or monitoring the interaction between agents.

While immersive visualization has advanced significantly [3], the creation of a complex and dynamic virtual environment in the form of a multiagent system is not a trivial task as it combines technologies that are not equivalent. It is important for the modeller of a multiagent simulation to be able to focus on the model development and avoid the complexity of how to deal with setting up an immersive visualization. The ideal case would be to establish a connection between the simulation and the visualization interface using minor configurations. The connected systems would then automatically generate the 3D representation of the virtual world from the simulation output, while they would enable an immersive movement of the human observer in the simulation without the need for further adaptation. However, such a combination of systems introduce technical and conceptual issues which have to be tackled in a profound way, starting from a theoretical analysis. In this contribution we discuss challenges and their solutions for visualizing the dynamics of a multiagent simulation in a CAVE-based virtual environment [4].

The remainder of this work is organised as follows. Initially, in Section 2, we describe the system and its basic requirements. Then Section 3 analyses the particular challenges which may occur when realizing such a system which enables the immersive visualization of an agent-based simulation in a CAVE. Section 4 sketches alternative architectures for the coupled system while the involved challenges are elaborated. In section 5, we discuss our first attempt to create a prototype. In the remainder of this work we discuss the work related to our approach and finally we give a short conclusion.

II. System Description and Requirements

The overall goal of this work concerns the creation of a framework that will be able to visualize the dynamics of a multiagent simulation within the CAVE. Two are the main concerns towards the achievement of this goal:

1) The representation alignment of a 2D multiagent simulation with a fine grain 3D visualization platform.
2) Human immersion in one representation.

The idea behind this approach is depicted in Fig. 1. It is clear that in order to achieve these tasks it is important to successfully align non identical representations before enabling a human to join the simulation.

Our initial task deals with the coupling of two dynamic representations of the same multiagent system, on different levels of temporal and representational granularity. We consider that *simulation* and *visualization* are two different systems-platforms representing the same multiagent model. The simulation is not embedded into the visualization system, but rather represented separately. This coupled system should communicate explicitly the information related to any given state of the simulated situation, while consistency between the two representations has to be assured.

Simulation is a more qualified representation, with refined object structures. Contrarily, the visualization refers to the 3D representation of the multiagent system using detailed object models and complex animations. It is of great interest to note that the two systems have different characteristics and a clear

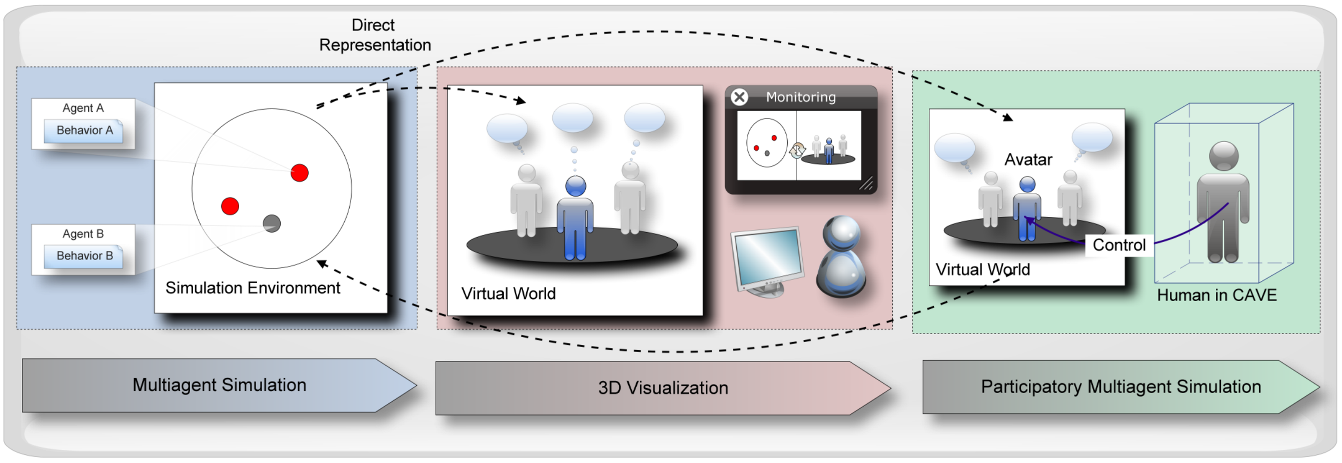


Figure 1. Graphical illustration of the components involved in the system and their relation. Multiagent simulation representation (*left*), 3D visualization of the same representation; enables better insight for the human modeller (*middle*), human inside CAVE and participatory multiagent simulation (*right*. The human has a direct representative in the simulation which he controls (avatar).

Table I
COMPARISON OF VISUAL ASPECTS BETWEEN MULTIAGENT SIMULATION AND VIRTUAL WORLD

	Simulation	**Virtual World**
Role of visualization	Add on for illustration	Raison d'etre
Spatial Dimensions	2D/3D, discrete/continuous	3D continuous
Temporal Resolution	Arbitrary	Real-time
Viewpoint	Bird-eye view	Camera entities
Representation Detail	Coarse	Observer-camera distance
Object Models	Points, polygons	Detailed 3d models
Behaviour Models	Complex behaviour	Complex animations
Interaction	Parameter adaptation	Directly with characters

separation of concerns. To indicate an example, one could think of a large scale pedestrian simulation [5] against the virtual world crowd simulation [6]. The simulation aims to represent the behavioural aspects of the modelled system in the simplest but still valid way, whereas in visualization the focus lies on the realistic representation of the 3D object models and their animated behaviour. Table I, gives a brief comparison of the two types of representations based on visualization features.

It is also worth mentioning that both systems should run in a both coupled and decoupled way, on the same or even on different computers. Consequently modularization and online data generation should be placed among the tasks that the final system should accomplish. Based on these functional requirements it is clear that the solution must be more than just connecting agents and environments to a number of specialized default object models and animations, using default infrastructure feeding a preconfigured virtual environment. A generic solution should be developed, that means it should work for different models in several domains.

In the next phase, the focus lies on the immersion of a human in the system. The modeller can join the simulation for instance as an observer, looking through the eyes of a particular agent. In this way the human can see what the agent perceives and how it reacts as a unit. In addition, if the simulation enables participation (i.e., participatory simulation [7]), then the human can interact with the individual agents in the environment and access information related to their behaviour. The variety of possible interactions is much higher than in macro simulations where only one observation point is possible.

It is evident that developing a framework capable of incorporating such functionalities would offer better insight and it could lead to concrete assessments over the plausibility of the simulation model. However, despite the fact that the change of perspective is a central idea for face validation, there are still several technical as well as conceptual challenges beyond the mere motivation.

III. CHALLENGES

A. *Representation Alignment Issues*

Focusing on the technical level of this task that is to map abstract concepts in simulation to graphical assets in visualization, one can clearly identify the difference in the level of abstraction. The exact representation of the simulated multiagent model in both systems is clearly inefficient. It is necessary to define a formal way to communicate specific information that is related to the entities and their actions between the two platforms. Communication in that sense is not only the generation and transmission of data but also the interpretation of it (eg. Agent generation: Female/ Male, Activity: Walk/Run/Idle etc.).

An additional complexity comes from the different temporal resolutions of the two representations. Synchronisation and consistency during runtime are important in the efficiency of the overall result. We organize the issues involved according to the following two dimensions: Modelling phase and Runtime phase.

1) Modelling Phase:

a) Mapping agents and entities to object models: Multi-agent simulations with a spatial environmental model, with a few exceptions, are based on 2D representations. In contrary, characters in the virtual world are presented at a higher level of detail, involving realistic object models. Fig. 2 illustrates such an example. This gap has to be bridged, thus it is important to establish a method to efficiently associate agents to their corresponding object models. This entails an amount of information about the object model such as shape, size, scale in order to plausibly visualize the simulated objects/entities. The measures must be corresponding. For example, if a 2D shape is used in collision avoidance, the bounding box of the 3D object model in the virtual world should not be much larger in order to maintain the effects of the collision avoidance behaviour. Hereby, the visualization of entities (i.e., environmental resources) in the virtual world may be more problematic and time consuming for the modeller than expected. That is because the virtual world may involve a rich environment with a large number of heterogeneous passive entities such as obstacles with different shapes, materials and textures, which in simulation may be visualized by the exact same 2D shape.

b) Mapping agent behaviour to animation: The behaviour of the agents should also be depicted. In order to have a concrete visual representation, not only the geometries of the object models are necessary but appropriate animations as well. Detailed animations have to be configured (e.g., movement speed, frame rate of display, etc.) and to be connected to the relevant information in the simulation model which may correspond to abstract notions. Additionally, if the internal state of the agent is changing during the simulation run, the animations and geometries should also follow. For instance, assume that we want to visualize the life cycle of a human agent. Morphing operators should be used in order to enable changes in the shape of the object model. A problem rises when internal states that are present in the simulation, do not have any corresponding visual representation in the virtual world. To indicate an example, consider the case in which an agent turns red when a value is below a specific threshold (eg., when the agent is hungry).

Apart from the dynamics driven by the individual behaviour of the characters, the interactions between the agents have to be considered as well. This means that dynamics led by the full process of agents' interaction become a critical issue for visualization (e.g., characters that talk, wave to each other etc.). Additionally, on the simulation side, only aspects of an agent that affect the other agents' behaviour need to be visible to the modeller, whereas in the virtual world only what is within the observer's field of view is supposed to be visualized. In this case, the level of detail (LOD)[8] plays an important role in the representation of dynamics.

In the same context, the global environmental properties have also to be dealt. A multiagent simulation may involve dynamics that are triggered by special environmental entities which effect globally the overall visualization (e.g., evacuation signal). Then the question of what should be visualized, where and how seems critical.

c) Rendering and configuration: The virtual world needs additional configuration in order to provide a rendered scene that preserves realism. Infrastructure such as cameras and lighting has to be adapted according to the demands of each scenario.

2) Runtime Phase: Visualization in the visualization system is real-time, measured in frames per second (fps) whereas in simulation the update is arbitrary. This is a problem that has to be tackled. In many applications, simulated time is intentionally different from real time so as to facilitate the testing procedure. Essentially, the simulation time is as fast as it can be (i.e., faster or slower when compared to real-time).

Assuming that the visualization is following the more qualified simulation, we can identify two problematic cases:

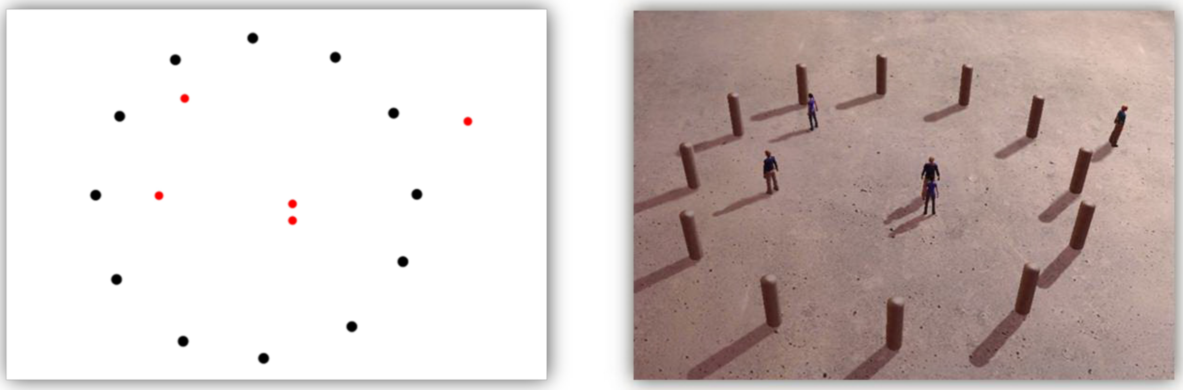
- The visualization system is slower than the simulation
- The simulation is slower than the visualization system

a) The visualization is slower than the simulation: This means that the visualization cannot afford the information flow. If we consider that the update difference is not that large, then a possible reason for this problematic situation to occur is that the characters have more elaborated object models whereas their corresponding agents have simple reasoning structures.

b) The simulation is slower than the visualization: In this case, the simulation is not able to feed the visualization with data in a timely manner. Depending on the update difference, it is possible that the visualization may have to stop and wait for new incoming data and eventually no real-time visualization is possible. A reason behind this problem could be that the agents hold costly reasoning structures while the characters in the visualization have simple object models. Additionally, in the simulation, there is not stable need for coordination or planning in every reasoning cycle. This may lead to quadratic or even exponential agent update whereas in pure rendering is linear in the number of characters.

B. Human Immersion Issues

The idea of involving a human in one representation can influence in a significant way the overall system's operation as this immersion will have an effect on the behaviour of the rest of the agents. In a simple case, the immersed human-avatar should be perceived as an obstacle for the other agents-characters walking around its position. However, if from being simply an observer, the avatar is interacting with other agents, then both its actions and their results have to be transferred back to the qualified representation and to be integrated to the running simulation so that the corresponding agent actually



(a) 2D simulation model (b) 3D virtual model

Figure 2. Illustration of a simulation model in contrast to its visualization in virtual space

executes the actions induced by the human. As a consequence the system appears with mixed qualifications resulting to a significant increase of complexity. This type of coupling that is bidirectional, can cause an evident problem for the synchronisation; simulation and visualization time must be in line. One of the two platforms has to take over the control with respect to the time advance so as to assure consistency during the simulation run.

After analysing a number of key issues, the remaining question is how such a system could be efficiently realised so as to ensure that the focus will not be shifted from the multiagent simulation and without adding any significant effort to the modeller. These challenges need to receive full attention in our work. In the following section, we are elaborating the problems and discuss potential solutions for coupling the two representations through conceptual architectures of the overall system.

IV. Architecture Alternatives

A. Architecture A

Simply sending information from the simulation about (dynamic) positioning of the agents to the visualization platform is not sufficient. Due to the different resolution and granularity, such a direct one way connection wouldn't solve the overall task, as information about the current state of the simulated situation (agents, entities, global properties etc.) has to be transferred and processed on the visualization side as well. Moreover, if we assume that the visualization engine is powerful enough to render the scene, an information overflow or lack of data (depending on the simulation time) is very probable to occur and there is no automatic way to avoid it. The most critical problem lies on the fact that this approach does not provide full functionality. The human can only be an observer when connected to the visualization platform. Therefore, we propose an architecture that is capable to handle the transfer of all information between the two systems. Fig. 3 depicts schematically the framework architecture. The basic elements in this conceptual view are:

- Simulation layer
- Visualization layer
- Networking component
- Simulation Visualizer component
- Control layer
- Human in CAVE

1) Simulation Layer: This is the layer in which the multiagent simulation model runs. Apart from the responsibility to drive the system's dynamics, in this side the generation of the scene takes place too. The starting situation of the simulation is stored and managed here. The situation scene should be exported in a format that can be (automatically) imported to the visualization platform.

2) Visualization Layer: In the visualization side, the scene has to be configured. Every agent in the simulation and every entity, should correspond to the relevant object model in the virtual world. Additionally the rendering features have to be configured as well. Lights, cameras have to be set up in a way that the scene can be rendered properly. The questions of how many lights are necessary, or where should they be positioned so that the scene is appropriately illuminated, rise. There are a number of algorithms from graph theory that can be used for automatically solve this problem [9]. The camera's configuration can be set to a standard parameter while the speed of the camera movement can be adapted to a reasonable value considering the size of the overall scene.

3) Networking Component: The simulation and visualization platforms, are coupled using a client/server communication bus in order to send messages containing respective information from one system to the other.

4) Simulation Visualizer Component: Assuming that the basic visualization information shall only be handled within the visualization system keeping the actual multiagent simulation model clean from such information is important. Consequently, the question how to import the information to the visualization platform where no ontological information about the original model might be available arises. To cope with

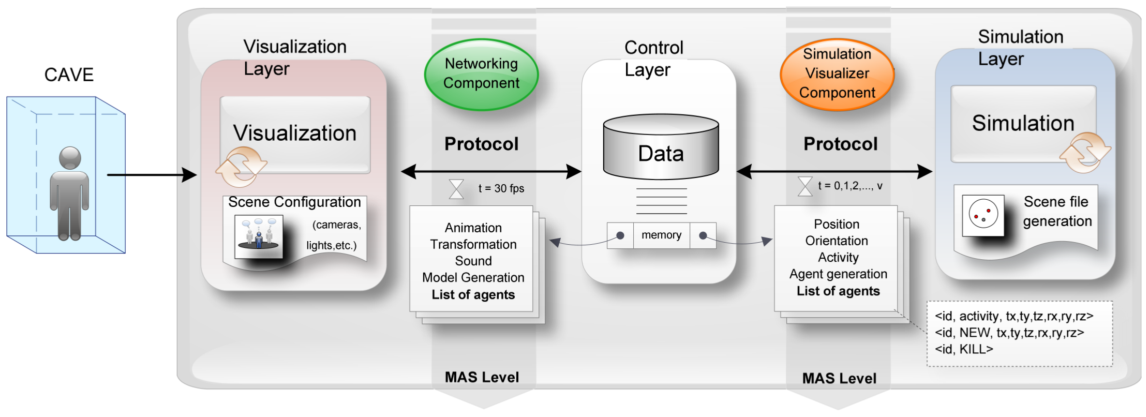


Figure 3. Schematic illustration of the framework architecture A

this problem, we consider the implementation of a component that keeps track of the simulation dynamics in an abstract, yet sufficiently detailed way.

This component is responsible for establishing a connection on the (abstracted) context between simulation and visualization. To be more specific, in every update cycle of the simulation, for every agent, information about its current state (e.g., position, orientation, activity etc.) is sent to the visualization platform. This information can contain the agents id, position, rotation, activity etc.

Each activity token corresponds to an animation that is displayed until a different token for the agent with the given id is received. The character connected to the given id is moving to the new position and orients itself according to the given rotation information. Accordingly, a reserved token could generate or delete agents from the visualization.

Finally, regarding the global properties, a specific message from the simulation may be connected to a particular sequence of changes in the visualization, as for example a signal ”fire” might be interpreted as the trigger coming from the global simulation entity world for visualizing smoke in the virtual world.

5) Control Layer: Until now we have dealt with the visual representation problems but the synchronization problems still remain. In the case where the visualization system is slower than the simulation, there are several possibilities for dealing with this situation. Clearly, the easiest solution is to reduce the simulation time advance and slow down the simulation. As we assume no inherent connection to the real-time, the simulation time can be reduced. Another alternative could be that the visualization system saves all the incoming information for each character and then renders the events in the correct sequence. Yet, if the buffer of activities is restricted then problems might occur. Information has to be skipped causing gaps in the visualized information. Of course such a solution should be avoided as it wouldn't support the validation of the multiagent simulation model. In the second situation where the simulation is slower than the visualization, the problems are more critical. As the simulation is unable to feed the visualization in a timely way, one solution could be that the simulation data are being recorded and visualized offline otherwise the visualization has to stop and wait for new data to arrive before proceeding to any action, which is clearly confusing for a human observer.

Another issue to be taken into account, is the granularity of control. We have a qualified representation and a following one. However, the more realistic the visualization shall be, the more the visualization engine will need to take over the control of the details of the interacting characters. Detailed animations with different configurations, morphing, sound, etc. are concepts that are not available in a simulation engine, the question is whether the information for planning and configuring their usage is handled on the simulation side or the underlying reasoning is done in the intermediate components on the visualization side. The information from the simulation has priority, but if there is not sufficient detail, then the visualization engine has to compensate. Therefore, which of the two platforms takes care of which details?

Due to these problems, we introduce in this layer a module that is capable of managing the flow of information between the two systems with a certain degree of reasoning while taking into account the problem of having different temporal resolution. When information arrives arbitrarily from the simulation, the particular data is stored so as to be used when necessary in the visualization engine. Then, if there is a conflict identified, it is the control layer responsible to find an intelligent way to resolve it. For instance simplifying the path of the characters when a number of positions has been received from the simulation that could not yet be visualized, deciding which agents are displayed on which level of detail. When the simulation is slower than the visualization, the representation component may extrapolate the behaviour of the agents based on an estimation where the next steps of the agents may be oriented towards.

6) Human in CAVE: Taking into consideration the architecture proposed, an asynchronous operation of the two systems would imply the need to have a roll-back function in the simulation side. Imagine a case in which the simulation is

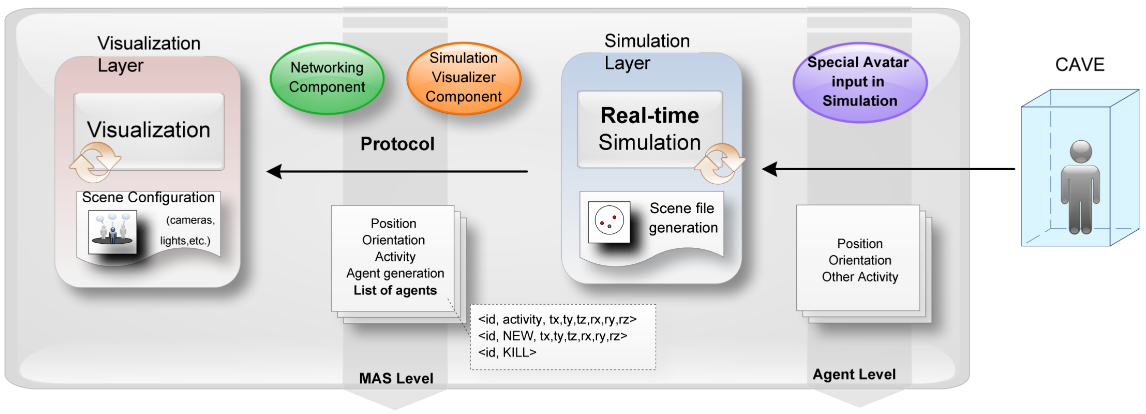


Figure 4. Schematic illustration of the framework architecture B

processing the agents state in $t_S = 100$ while in the user connected in the visualization platform, is changing the flow of actions in $t_V = 10$. The simulation has to adapt to this change and continue from $t_S = t_V = 10$. Nevertheless, despite the increase of complexity, the active human immersion in the system appears very attractive and with great potential in the process of evaluating the plausibility of a model and thus such prospect has to be considered in future work.

B. Architecture B

The second alternative is depicted in Fig. 4 and proposes a completely different architecture. Main characteristic in this schema is the involvement of a human user in the more qualified representation; the simulation layer. In this case the human controls one agent, perceives what the agent may perceive and manipulates the simulation through the agent's effectors.

The simulation layer is enhanced by the use of a *Special Avatar Input Component*. This component enables the human immersion in the agent-based simulation. Information is sent via the sensors in the CAVE affecting the models with primitive calls.

In this schema, we still have to deal with all the problems of configuring the simulated scene in both platforms (visual representation issues) similarly to the previous architecture. The *Simulation Visualizer Component* similarly to architecture A, receives the information from the simulation platform and visualizes it an appropriate way.

Central to this approach is that the simulation has to be adjusted so that it runs in real-time. The time resolution issue between the two platforms is hence eliminated and there is no need for a *Control Layer* as in the previous architecture. In a technical level, a time advance that is similar to real-time is advisable for validation purposes as this is what is the least confusing or distracting the human observer from the dynamics.

An important parameter that should be taken into consideration in the proposed architecture is that the visualization is depending on the simulation in order to start. This means that the visualization system is not treated as an active platform but it rather plays the role of an external visualization component upon the simulation which doesn't need to hold any status. In addition to that, the functionality of the overall system, is totally based on the simulator used and the generic character of the framework is totally depending on the protocols used.

V. Prototype

In an initial attempt, we tried to realize the generic coupling of two such systems in a prototype. Our work was grounded by the use of SeSAm[1]; a modelling and simulation platform and the Horde3D[2] GameEngine.

It uses client-server communication for transmitting information about the visible effects of agent actions from the simulation to visualization. As depicted in Fig. V the server maintains the overall scene and updates are sent by the simulation client. The scene is originally created within the simulation and then transformed into the game engine scene format. To do so, we developed an export function that generates a description of the visualization scene out of the simulated situation. Tokens replace object models, that means pointers to the object models have to be inserted manually. However, as long as there is no automatic generation of geometries from the simulation engine, this export is restricted as it assumes that appropriate object models are existing. A future version must either generate appropriate object models or must be able to scale object models based on the precise information from the simulation situation. In the latter case, simulation entities might have to be augmented with a height value describing the scaling. The protocol used here includes each relevant entity's position, orientation and animation in each update. It also contains events that are sent only once, such as changes in the environment or the generation or removal of agents from the world scene.

The next step was to bring the Horde3D GameEngine to an immersive virtual environment where a human can observe the simulation model from a very close point of view. To accomplish this task, a pre-existing multiplayer architecture

[1]SeSAm: http://www.simsesam.de
[2]Horde3D GameEngine: http://hcm-lab.de/projects/GameEngine

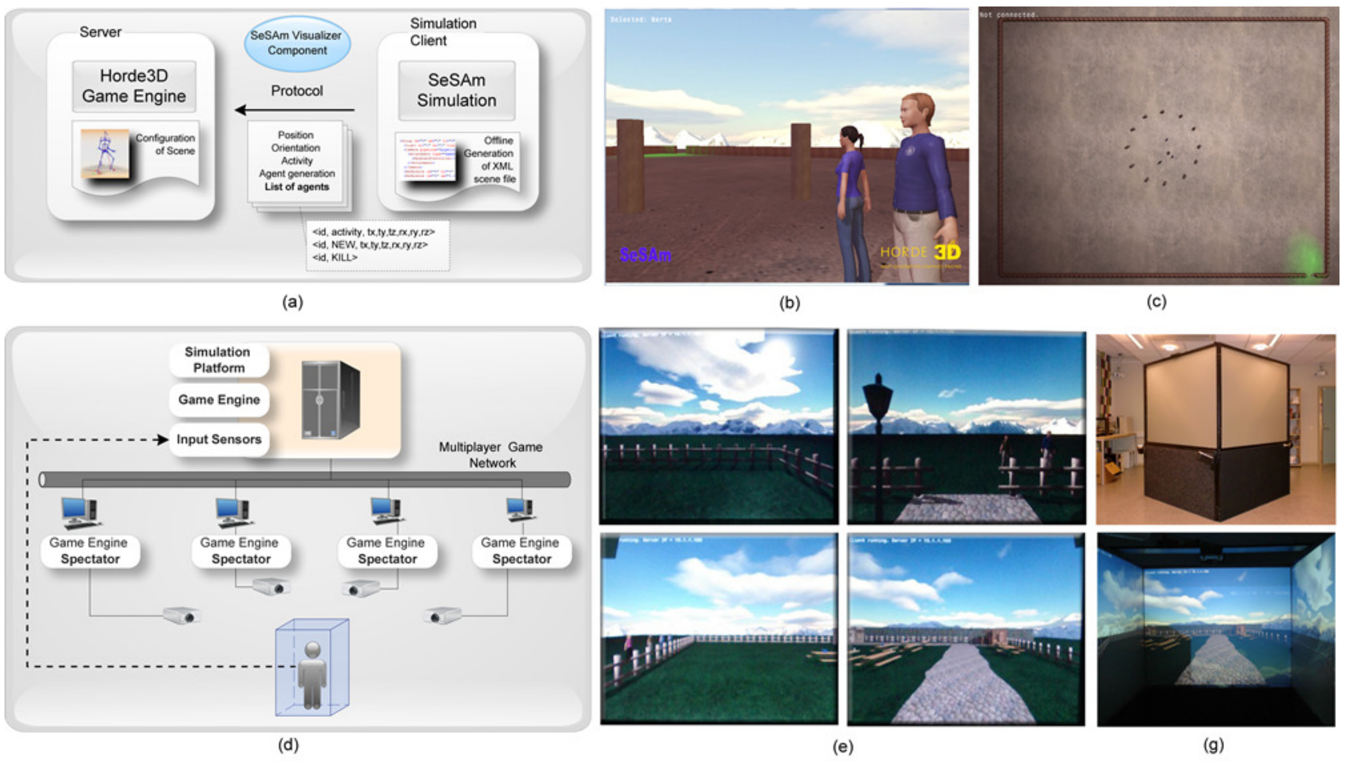


Figure 5. Prototype aspects. (a) Initial architecture; SeSAm-Horde3D GameEngine coupling, (b) Observation of dynamics through agent's perspective, (c) Visual representations of the same multiagent model as seen from bird's eye view, (d) Horde3D GameEngine in the CAVE, (d)The four "spectator's view", (e) CAVE (*upper*) and scene projected in the CAVE (*down*).

within the GameEngine was utilised in order to project the simulated situation to the CAVE. The protocol in this multiplayer architecture is state-based, i.e. the updates are sent at regular intervals, each containing the status of every relevant agent or entity in the scene. We use one server and five clients. The server hosts the virtual scene and is responsible for distributing changes happening in that scene to the clients. Four of them simply render the scene provided by the server in each frame without providing input on their own. This mirrors the concept of "spectators" in multiplayer games and these four clients are used to provide the data for each of the CAVE's projectors. The fifth client is connected to the CAVE's sensors and provides the sensor input for the human to navigate in the scene.

VI. Related Work

A. *Multiagent systems, visualization and 3D virtual worlds*

Multiagent models in physically simulated 3D worlds are popular since the seminal evolving creatures of Karl Sims [10]. Later developments in microscopic pedestrian simulation produced an increasing need to provide 3D spatial representations as a visualization method on generic multiagent simulation platforms such as Repast[3] or MASON[4] which embedded Java3D into their simulation platforms. Similarly, game engines where used but mostly as a mean to model the environment of the agents, while sensor data and action commands are communicated between the agent and the game engine [11]. Contrarily, in our case, the game engine is responsible only for visualizing the dynamics of the simulated situation that is apparently modelled in an external simulation platform. The combination of virtual worlds and simulation is prominent in crowd simulation as well [6],[12]. Main consideration in our approach is the generic character of the solutions thus our system should not just be applicable for crowd simulations.

Korhauser et al. [13], give design guidelines for multiagent simulation visualizations, adapting general design principles about shapes and colours of the agents, grouping the entities for giving advice when visualizing agent system simulation dynamics. Our research is also related to early works on user interfaces for multiagent systems. Avouris [14] classify different multiagent system architectures for identifying special challenges in designing interfaces to multiagent systems - including multiagent simulations focusing on the bird's eye view.

Consistency plays a major role in our work. Thereby, a relation exists between the proposed problem and solutions from the area of distributed interactive systems such as mul-

[3]Repast: http://repast.sourceforge.net/
[4]MASON: http://cs.gmu.edu/eclab/projects/mason/

tiplayer network games, where consistency has to be assured for presenting the same situation to different users. Several techniques have been developed for avoiding or dealing with inconsistencies coming from latency and jitter [15],[16]. In our case, we identify the major problems to be the different resolutions and synchronization problems between two full representation of the same system, not a distributed representation. Nevertheless, the synchronization problem has similarities with the consistency problem of distributed interactive systems. Clearly we will have a closer look onto techniques of dead reckoning, etc.

The idea underlying our work is relevant to some degree with the principles of the Model View Controller (MVC) paradigm [17]. MVC design pattern have been widely used in Web applications which promoting the separation of visual presentation from logic. However our work has a core difference. The simulation platform is already separated from the visualization.

B. User involvement in agent-based simulation

Our vision is similar to Repenning and Ionannidou [18] who aim at enabling an end-user to create complex visualizations. They are proposing a tool that facilitates the distortion of existing object models for creating process visualizations in an accessible way. They are apparently addressing the same step in a simulation visualization process that is directed by professional in animation programs such as Maya or 3D Studio Max[5].

Moreover, several methodologies on how to incorporate human actors in large scale simulations with autonomous agents are present in literature [19],[20]. In most of the cases, the agents are controlled by a human sitting in front of a computer. However, we consider a full immersion of a human in a CAVE. Nevertheless, in our approaches the main consideration is the model validation and the evaluation of the plausibility.

VII. Discussion and Future Work

In this work we analysed the challenges which have to be solved when the dynamics of an agent-based simulation are visualized in a CAVE. An immersive face validation complements the usual data-driven validation on the macro level due to the fact that it allows to check the plausibility of individual behaviour trajectories. Several architectures were also discussed with the main goal to frame the design process towards the realization of the intended system.

Our future work is oriented towards the collection of the building blocks for a modelling language that supports the generic connection to animations. The testing and evaluation of the deployed system is also an important aspect of the coming work. Tests are going to be performed in a simple scenario (eg. evacuation or flocking model) with different numbers of agents/characters ranging from 5 to 100 individuals.

[5]Autodesk: www.autodesk.com

VIII. Acknowledgements

This research work is part of the ”Human-in-the-Loop Modeling and Simulation” project funded by MINNOVA and whose support the authors gratefully acknowledge. We also thank Augsburg University for offering their Horde3D GameEngine used in this work and particularly Michael Wißner for his valuable help.

References

[1] O. Balci, “Validation, verification, and testing techniques throughout the life cycle of a simulation study,” in *Simulation Conference Proceedings, 1994. Winter*, dec. 1994, pp. 215 – 220.

[2] F. Klügl, “A validation methodology for agentbased simulations,” in *Proceedings of the 2008 ACM Symposium on Applied Computing*, R. L. Wainwright and H. Haddad, Eds. ACM, 2008, pp. 39–43.

[3] T. Yapo, Y. Sheng, J. Nasman, A. Dolce, E. Li, and B. Cutler, “Dynamic projection environments for immersive visualization,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, june 2010, pp. 1 –8.

[4] H. Lee, Y. Tateyama, and T. Ogi, “Realistic visual environment for immersive projection display system,” in *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on*, oct. 2010, pp. 128 – 132.

[5] F. Klügl and G. Rindsfüser, “Large scale pedestrian simulation,” in *Proceedings of MATES 2007*, ser. LNAI, J. Müller, P. Petta, M. Klusch, and M. Georgeff, Eds., vol. 4687. Springer, 2007.

[6] N. Pelechano, J. Allbeck, and N. I. Badler, *Virtual Crowds: Methods, Simulation, and Control*. Morgan and Claypool Publishers, 2008.

[7] M. Berland and W. Rand, “Participatory simulation as a tool for agent-based simulation,” Setubal, Portugal, 2009, pp. 553–7.

[8] M. Wißner, F. Kistler, and E. André, “Level of detail ai for virtual characters in games and simulation,” in *Proceedings of the Third international conference on Motion in games*, 2010, pp. 206–217.

[9] *3D Game Engine Design, Second Edition: A Practical Approach to Real-Time Computer Graphics*. The Morgan Kaufmann Series in Interactive 3D Technology.

[10] K. Sims, “Evolving 3d morphology and behavior by competition,” in *Artificial Life IV Proceedings*, R. A. Brooks and P. Maes, Eds., 1994, pp. 28–39.

[11] E. Norling, “Capturing the quake player: using a bdi agent to model human behaviour,” in *Proc. of AAMAS’03*. New York, NY, USA: ACM, 2003, pp. 1080–1081.

[12] D. Thalmann and S. R. Musse, *Crowd Simulation*. Springer, 2007.

[13] D. Kornhauser, U. Wilensky, and W. Rand, “Design guidelines for agent based model visualization,” *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 2, 2009.

[14] N. M. Avouris, “User interface design for DAI applications,” in *Distributed Artificial Intelligence: Theory and Practice*, N. M. Avouris and L. Gasser, Eds. Kluwer Academic Publisher, 1992, pp. 141–162.

[15] C. Diot and L. Gautier, “A distributed architecture for multiplayer interactive applications on the internet,” *Network, IEEE*, vol. 13, no. 4, pp. 6 –15, jul/aug 1999.

[16] D. Delaney, T. Ward, and S. McLoone, “On consistency and network latency in distributed interactive applications: a survey–part i,” *Presence: Teleoper. Virtual Environ.*, vol. 15, pp. 218–234, 2006.

[17] A. Doray, “The mvc design pattern,” in *Beginning Apache Struts*. Apress, 2006, pp. 37–51.

[18] A. Repenning and A. Ioannidou, “End-user visualizations,” in *2008 Int. Conf. on Advanced Visual Interfaces (AVI 2008), Napoli, Italy*. ACM Press, 2008.

[19] P. Guyot and A. Drogoul, “Multi-agent based participatory simulations on various scales,” vol. 3446 LNAI, Kyoto, Japan, 2005, pp. 149 – 160.

[20] T. Ishida, Y. Nakajima, Y. Murakami, and H. Nakanishi, “Augmented experiment: Participatory design with multiagent simulation,” in *Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, 2007.