Bi-directional Coupling of Simulation Tools with a Walkthrough System

Bengt Mueck [1,2]
Heinz Nixdorf Institute
University of Paderborn
mueck@hni.uni-paderborn.de

Wilhelm Dangelmaier [1]
Heinz Nixdorf Institute
University of Paderborn
whd@hni.uni-paderborn.de

Matthias Fischer [3,4]
Heinz Nixdorf Institute
University of Paderborn
mafi@uni-paderborn.de

Wolfram Klemisch [3]
Heinz Nixdorf Institute
University of Paderborn
wolfk@uni-paderborn.de

Summary

Visualising is a method used to help experiencing and understanding causal cohesions in simulation processes. For this purpose, tools for visualising are already implemented in prevalent simulation systems. The user creates his simulation model and generates a 3-dimensional (2,5-dimensional) visualising by means of the simulation system. This helps examining the process which makes it easier for the viewer to "understand" it. Simulation tools usually only provide the opportunity for a unidirectional visualising. In a 3-dimensional surrounding the viewer can not implement an interaction with the simulation while the system is running. Though an interaction during the simulation run enables the user to gain a better understanding of causal cohesions. Solutions via HLA are sophisticated and therefore rather suited for extensive projects.
We present a distributed system consisting of a commercial manufacturing simulation tool, a coupling module and a walkthrough system. The distributed system in conjunctions with the coupling module guarantees generality and a wide field of applications of the walkthrough system. Further it guarantees flexibility and selection of the specialized graphics hardware for the walkthrough system. A further contribution of this paper is the solution of the time synchronisation problem caused by simulation tool and walkthrough system.

1 Introduction

1.1 Problem

A visualising of simulation processes for the viewer facilitates his understanding and experience of simulation processes. An interaction between him and the system involves the viewer more directly into the process and thus provides for a better understanding.

---

[1] Heinz Nixdorf Institute, Business Computing, esp. CIM, D-39095 Paderborn, Germany
[2] Partially supported by DFG grant DA 155/24-2
[3] Heinz Nixdorf Institute, Theoretical Computer Science, D-39095 Paderborn, Germany
[4] Partially supported by DFG grant ME 872/8-1 and the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT)

Most commercial manufacturing simulation tools [KlG99] allow for a visualising of the run but not for an interaction. The call for an interaction during the simulation process of a conventional manufacturing simulator causes many problems which will be discussed in this paper. Furthermore, we develop a solution architecture, implement it and introduce the results.

1.2 Demands on the Solution

To generate no proprietary specialised solutions the system should be as flexible as possible, thus it should be possible to employ various simulation tools.
Before the simulation is carried out a model has to be created which usually does not contain any information on the visualising. In order to guarantee an efficient handling of the entire system it is fundamental to generate a visualising model out of the simulation model. During the creation of the simulation model it should be feasible to implicitly model positions and movements of the objects in the visualising.

1.3 Time Synchronisation Problem

If simulation and walkthrough system run independently they both operate within their own time scheme. This implies certain requirements: Events in the simulation may trigger visualising events. Those can not be pictured accurately, if they are happening in the past of the walkthrough system-time. Consequently, walkthrough system-time may never outrun simulation time. If interactions are triggered in a walkthrough system, it is only possible to transfer them into the simulation at the current point of simulation time without having to reset simulation. The viewer has to wait adequately long for a reaction in the visualising, if simulation time is significantly faster than walkthrough system time. Hence, simulation time may not be considerably faster than walkthrough system time in order to guarantee the feeling of a direct reaction. We will deal with two procedures to solve this problem. One of those has been implemented and examined in our prototypic system.

1.4 Related Work

The coupling of simulators and visualising systems via HLA [DMS99] [Jes01] both pursue similar aims. HLA offers an extensive range of functions but is rather sophisticated in its use. With only few exceptions [LSU99] HLA is not widely accepted in the world of production simulators [KlGu99]. The concept that we are discussing in this paper is less comprehensive, smaller and specialised on the bi-directional coupling of simulators within walkthrough systems.
Time synchronisation problems are of central importance for the implementation of 3-dimensional Multiplayer-Games. Examples are Doom, Duke Nukem, Ultima Online, Unreal or the ego-shooter Quake [Abr97]. In the beginning the simulation in games like Doom or Duke Nukem was carried out strictly synchronous (peer-to-peer) [Swe99]. Later on the need for a broader scalability caused a transition to client-server technology in games like Ultima Online and Quake [Abr00]. The game, including its current player positions and actions, is simulated on a central server while the simulation dates are broadcasted to the clients. In later versions of Quake simulation and prediction logic were

used on the client side [Swe99], which accounted for inconsistencies between the prediction of the client and the simulation of the server. To prevent this kind of inconsistencies we did not integrate prediction and simulation logic into our walkthrough system. This guarantees a walkthrough system which is rather universal and not dependent on the deployed simulator.

2 Architecture

The entire system provides two focal functionalities for the user. On the one hand, he has to be able to create models. On the other hand, he needs to "experience" the simulation interactively. Therefore we should speak of two independent users (modeller and viewer):

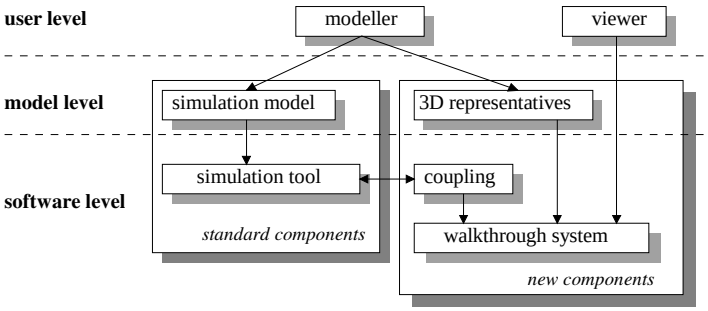

Figure 1: Architecture of the entire system with a view on each level

The viewer (see fig. 1) experiences and interacts with the system via a visualising component. He does not have access to any other user interface. The modeller utilizes tools to create models for the simulation as well as for the 3-dimensional visualising. In our current concept he will generate simulation models in a simulation surrounding. For the visualising simulation objects are attributed to 3-dimensional representatives (see fig. 1). Those representatives will be created by conventional modelling tools for 3-dimensional models. Simulation objects and their 3-dimensional representatives are strictly separated. The visualising system consists of two additional software components, which are the coupling module on the one hand and the walkthrough system on the other hand (see fig. 1). Both components represent individual processes. As individual modules may communicate via TCP/IP they can be implemented on various computers.
Through a strict modularisation and a communication among the individual parts which embraces all computers, diverse hardware can be employed. For instance, this architecture allows for the implementation of specialised visualising hardware. In the following we will discuss the tasks and functionalities of each component.

2.1 Tasks of the Simulation Tools

The simulation tool conducts the dynamic calculations which have been specified through the simulation model. Actions (e.g. movements of objects) within the walkthrough system are stimulated by those instruments. On the other hand, the simulation tool has to integrate user interactions of the walkthrough system into the

simulation within running time. The simulation tool does not only support the execution but also the creation of simulation models. In this model we consider the 3-dimensional composition of objects while modelling in order to enable fast and easy work with the system (the model should at least contain the position of objects on one plane). We therefore currently employ 2-dimensional coordinates of the simulation objects within the simulation tools. The 3-dimensional scene is created by the coupling module and the 3-dimensional representatives which have been produced before.

2.2 Tasks of the Walkthrough System

The walkthrough system represents the interface between the entire system and the viewer. It visualises the simulation for the viewer and picks up his interactions. The interaction has two implications for the walkthrough system:

- **Navigation through the scene:** The viewer should be able to move freely within the visualising in the simulation, which means he should be able to move towards various places and turn and move the whole scene.
- **Actions for simulation objects:** The viewer should be able to influence the simulation procedure by an aimed "picking" of simulation objects. It should, for example, be possible to move a switch on a conveyor belt. The walkthrough system reports to the coupling module which action or object the viewer has chosen. The coupling module then decides which actions to take.

2.3 Tasks of the Coupling Module

The coupling module provides for the connecting infrastructure of the individual parts. All communication between the walkthrough system and the simulation tool will be modularised here. It capsules specific communication interfaces between the simulation tools and the walkthrough system and furthermore undertakes a bi-directional transformation of messages into the appropriate language. Neither the simulation system nor the walkthrough system have to be altered for the respective partner. It is possible to simply connect and trigger events from out of the simulation. The capsuling of the coupling module seizes specific characteristics of the simulation as well as of the used simulation tools:

- **Simulation tool:** in the coupling module specific characteristics of the protocol and the parameters of diverse simulation tools can be hidden from the walkthrough system and pictured on an universal interface.
- **Simulation:** simulation-specific characteristics like, for example, the shift of a switch require a feedback to the simulation tool. The decision if an object should deliver a feedback or not should be taken without considering the picking of the object in the walkthrough system within the coupling module.

This will result in a strict separation of functionalities within the simulation system and the walkthrough system and will make the walkthrough system universally applicable.
An interface which has been formally defined in each direction enables the exchange of both simulation tools and the walkthrough system. Open interfaces enable not only the

implementation of various simulation tools but also the implementation of diverse visualising tools. In the current realisation every simulation object establishes an individual communication channel to the coupling module. As opposed to this, there exists only one channel from the coupling module to the walkthrough system. The coupling module ensures a correct attribution of messages to the objects.

3 Modelling

3.1 Modelling within the Simulation

The simulation tools should not only determine the modelling of simulation models, but should also define the position of permanent objects within the visualisation. Therefore descriptions should be available in a 3-dimensional format. Those descriptions need to be identified through explicit identifiers within the simulation.
The employed simulation tool (currently EM-Plant) arranges the objects on a plain. The coordinates on this plain help calculating the coordinates for a 21/2 dimensional visualising. Valuable and intuitive models can be created, if the proportion of the 2-dimensional objects to its arrangement-plain is equal to the proportion of the 3-dimensional objects to its surface. To visualise further static objects, which are not directly involved in the simulation (e.g. the floor, walls, worker etc.) types have to be created, that can be visualised but have no influence on the procedure of the simulation.

3.2 Modelling within the Walkthrough System

The configuration of static objects will be performed as described above. The position of moveable objects will be calculated during the running time of the simulation. A certain amount of 3-dimensional representatives have to be retained for the objects. Those have to be formed with the help of conventional modelling tools (3d Studio Max, Softimage etc.) and should be copied from the basis into the walkthrough system if necessary.

3.3 Construction of a Static Scene

To transmit all static visualising dates to the walkthrough system after the modelling an initialising event will be send from the coupling module to all objects within the simulation. Each static object in the simulation then calculates its own position and transmits his coordinates in combination with his type (resp. the identifier of his 3-dimensional representative) to the walkthrough system. The walkthrough system employs these dates to establish a static representation of the scenes.

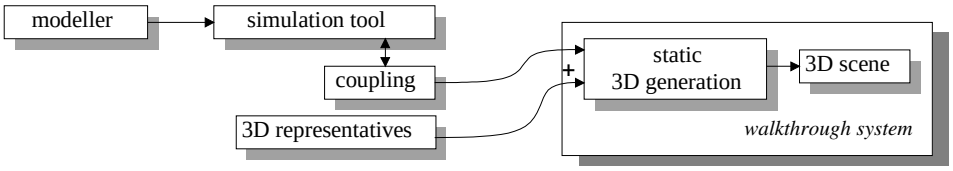

Figure 2: Construction of a static scene

Figure 2 shows the construction process of a static scene. With the help of the 3-dimensional representatives, which had been developed earlier, and the objectIDs, which had been delivered by the coupling module, the walkthrough system creates a static 3-dimensional scene. The coupling module receives the objects and its 2-dimensional coordinates from the simulation tool. The third coordinate is being added by the coupling module according to the specific simulation tool and the simulation. The entire process is executed before the actual simulation within its preprocessing and is summarized in the box below:

Steps for processing: *Static construction of the scene by the modeller*

1. Creation of a simulation description with a simulation tool
2. Creation of a 3-dimensional representative
3. Simulation-specific adjustment to the coupling module
4. Construction of a static scene of simulation tool, coupling module and walkthrough system

4 Bi-directional Coupled Dynamic Simulation

In this model we implement the loose coupling of simulation and walkthrough system, while both systems have separated instances of objects. Changes which have to be edited in both systems synchronously are mainly the production of new objects, the destruction of old objects and the movement of objects and user interactions which are directed at an object. Messages have to be defined at least for these operations.

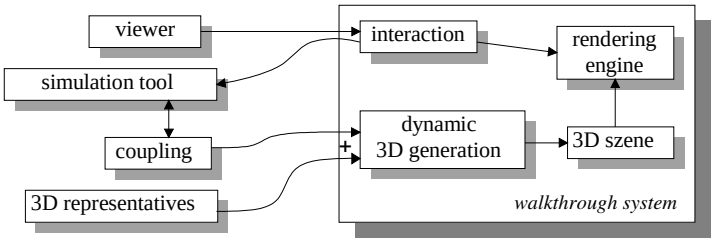

Figure 3: Bi-directional dynamic simulation procedure

The simulation module has to be able to create new objects within the walkthrough system. This necessitates the transmission of an identifier of a 3-dimensional representative (see fig. 3), its position and an identify for further communication about these objects from the simulation module to the walkthrough system.
If objects in the walkthrough system are to be destroyed, the identifier of those objects has to be transmitted to the walkthrough system. To move objects, starting-time, arrival-time and destination-positioning have to be broadcasted. It is also possible to convey the starting-time, the direction, velocity and the duration. The walkthrough system accepts user interactions and has to forward those (see fig. 3). This necessitates a transmission of objectID and the type of the interaction to the simulation. Further transmitted functionalities like e.g. information about the synchronisation of time are possible.

Message	Direction	Attributes
Create Object	Sim. -> Walk.	ObjectID, ID 3D Representative, Position
Destroy Object	Sim. -> Walk.	ObjectID
Move Object	Sim. -> Walk.	ObjectID, Starting time, Destination-time, Destination-position
User Interaction	Walk. -> Sim.	ObjectID, Type of Interaction

Table 1: Examples of some objects

4.1 Coupling Module

The coupling module collects messages, which stem from the connections to the objects of the simulation, and forwards those to the walkthrough system. Furthermore, the objectID helps distributing the messages about user interactions, which come out of the walkthrough system, through the adequate channels to the simulator. Our goal for the coupling module is to capsulate all simulation and simulation tools dependent tasks. Typical implemented tasks of our system are:

- Transfer of coordinates.
- Storage of the object coordinates, e.g. during movement of an object from source to destination.
- Storage of the object size for a right placing in the scene.
- Initialisation, creation, and deletion of moving objects. The simulation only cerate the start and end time of the object.
- Computation of the speed of moving objects.

All the communication will go through a TCP/IP connection (stream), which has been established once in the beginning of the simulation. Each package which has been send is equivalent to a produced message (e.g. “create object” or “move object”). It is 96 byte big and always contains the data-fields type of message, objectID and starting-time of the object in the simulation. Depending on the objectID data-fields like expiring-date, starting position, end position or rotation etc. may be added. For the receipt a thread has been implemented in the walkthrough system which is running parallel to the thread for the rendering of the walkthrough system.

4.2 Time Synchronisation

In concrete simulations simulation time jumps from one point of simulation time to the other. Calculations happen at one point of simulation time, though they need real time. If the simulation time is required to move forward in (resp. proportional to) real time, simulation time progresses in total have to be calculated faster than real time. Time within the simulators has to be stopped regularly. Visualising time should not outrun simulation time in order to prevent inconsistencies between visualising and the simulators. To compensate for the use of calculation time, simulators have to be given a forerun. Therefore progresses in the simulation can be visualised in real time=simulation time. A large forerun allows for the compensation of bigger variations in computation load, but prevents a fast user reaction to events. Varying fast reactions are sensible in different user scenarios. It would be advantageous to keep forerun time for these

experiments variable. If visualising time approaches simulation time crucially, a deceleration rather than a halt of the visualising helps the viewer to get a complete picture of the system. Visualisations are calculating due to the rhythm of the set-up of new pictures. The picture repetition rate of the visualising should be decoupled from the time progress of simulation.

Let t be the real time of our system. Furthermore $t_s(t)$ stands for the simulation time of the simulator and $t_w(t)$ for the walkthrough system time. With simulation tools there often occurs the problem that simulation time $t_s(t)$ is not linear to t, which means that it does not apply to all times t: $t=t_s(t)$. In many simulation tools walkthrough system time runs in a partly constant, stair-like function (see fig. 4).

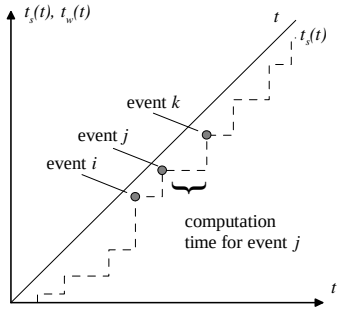

Figure 4: Real time and simulation time

The simulator will always try to keep close to real time t. This problem will intensify in a system with a walkthrough system which is bi-directionally coupled, as the entire system does not only include simulation time $t_s(t)$ but also walkthrough system time $t_w(t)$. This may cause problems because of various reasons:

- **Delay through transmission:** A walkthrough system which is running on another computer causes high transmission times. These may result in delays on t both computers, when simulation status or interaction specifications of the user are sent from one computer to the other.
- **Diverging of time bases:** The relative time basis of the hardware timer in both computers may diverge in the course of time and therefore complicate an exact synchronisation.

This means that it will get very hard to guarantee for all times: t: $t_w(t)= t_s(t)$. The term *follow-up time* Δt_{sw} stands for the time which the walkthrough time lags behind the simulation time. Correspondingly, Δt_w represents the time which the walkthrough system lags behind the real time:

$$\Delta t_{sw}(t) = t_s(t) - t_w(t) \qquad \Delta t_w(t) = t - t_w(t)$$

Follow-up times cause problems for the use of the simulator, if the simulation procedure is changing due to a specification of the viewer in the walkthrough system. Besides, it may get problematic if the progression of simulation time $t_s(t)$ is not clearly identified:

- **Too long follow-up time $\Delta t_{sw}(t)$:** If $t_w(t) < t_s(t)$ holds good, that is, if the walkthrough system time lags behind the simulation time, the action of the viewer may not be included into the simulation, as it is conflicting to future

events in the simulation which the simulator already has calculated. For instance, the viewer activates a switch for two conveyor belts. Nevertheless the next package will make its way over the old conveyor belt, as the simulation time had already advanced in such a way, that it had already been behind the switch and not in front of it, like it had been shown to the viewer.

- **To short follow-up time $\Delta t_{sw}(t)$:** If $t_w(t) < t_s(t)$ holds good, that is, walkthrough system time is running ahead of simulation time, the action of the simulator can not be simulated by the walkthrough system anymore, as the walkthrough system already has proceeded further.

The question which is emerging now is how to regulate walkthrough system time. A solution to both problems would be to reset simulation or walkthrough system for such timeframes. However, this would interrupt a fluent simulation and is therefore not always manageable. We will present two theories and discuss them.

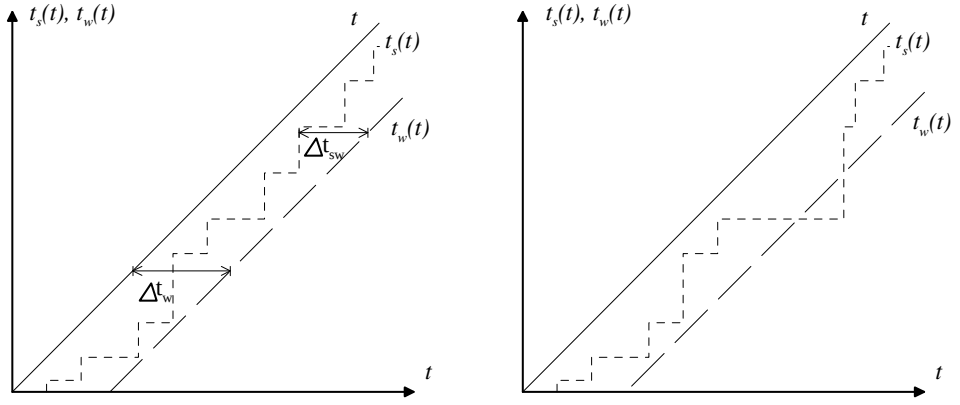

Figure 5: Walkthrough system with fixed follow-up time

4.2.1 Approach with Fixed Follow-up Time

On account of those problems, we will try to define follow-up time positively and as small as possible ($\Delta t_{sw}(t)>0$). A resolution which is easy to implement and which we have realized and examined in our system, implies the fixing of the follow-up time on a constant value. Communication costs t_c for the transmission of the first message from the simulation tool to walkthrough system could serve as an adequate estimated value, for example, since we have seen in our tests that those mainly dominate follow-up time.

To make sure that walkthrough system time will not outrun simulation time, we have added a tiny period of time t_o as a reserve, which means that the follow-up time of the walkthrough system equals:

$$\Delta t_w(t) = \Delta t_w = t_c + t_o$$

Walkthrough system time therefore has been moved to the right of real time by Δt_w .

4.2.2 Resolution with Adjustment

In our test we always have opted for a Δt_w which is big enough to guarantee $t_w(t) < t_s(t)$. In our applications the problems described above did occur rather infrequently, though

those problems will, of course, continue to be connected to the method of fixed follow-up time. To avoid the problems it might be sensible to choose another solution. It would be an alternative to readjust time $\Delta t_w(t)$ during simulation, so that $\Delta t_{sw}(t) > 0$ still holds true, but $\Delta t_{sw}(t)$ has to be kept as small as possible at the same time (see fig. 6).

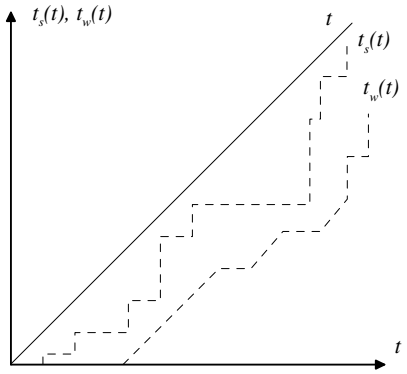

Figure 6: Walkthrough system time with readjustment

Naturally, this should happen smoothly in small steps. The time basis should be reduced or accelerated slowly. Otherwise the animation would result in too big jumps back or forth in the walkthrough system and would therefore trouble the viewer. We have implemented and examined the first procedure with constant follow-up time. The resolution with readjustment will be examined in future theses.

5 Results

In the method described above we have implemented and constructed a prototypic walkthrough system and a module for the coupling. In our test the module for coupling ran on the same computer as our simulation tool EM-Plant.

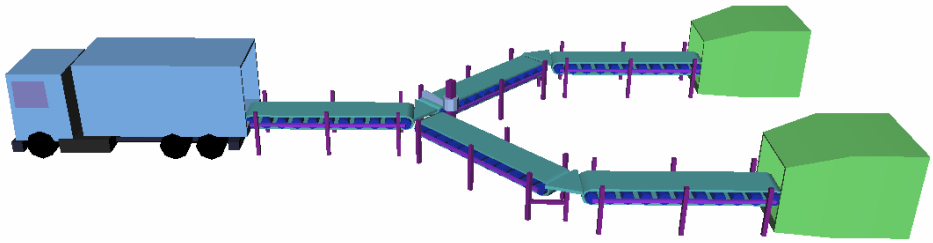

Figure 7: Employed simulation model

5.1 Employed Simulation Model

Modules for a commercial manufacturing simulation tool (EM-Plant) have been developed which responded to a fountain, an assembly-line, a switch and a swale. In the

walkthrough system the source has been modelled and presented by a lorry and the destination has been modelled and presented by a container (see fig. 7). The switch directed the packages from out of the lorry on two conveyor belts each towards one container.

5.2 Modelling and Static Scene Construction

These modules include those functionalities which have been developed above. Each object which has been formed out of those modules registers at the coupling module autonomously. After a stimulation the position in a 2-dimensional simulation model provides the data for calculating a 3-dimensional position in a walkthrough system. They just hand those dates over and thus generate a static 3-dimensional model.

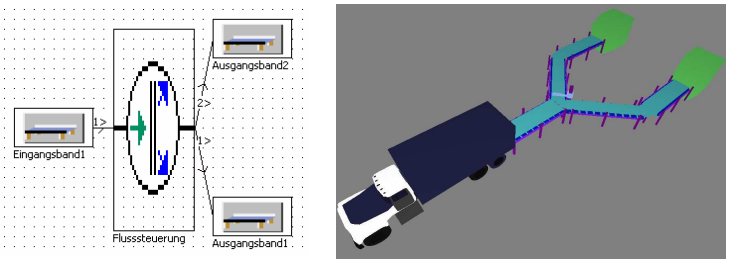

Figure 8: EM-Plant and walkthrough system after the creation of a conveyor belt

Figure 8 shows a model illustration of EM-Plant on the left and the 3-dimensional model which has been created by that in the walkthrough system to the right. The modeller of EM-Plant is not involved into the building of the scene after the creation of the simulation model. This will be triggered automatically by the walkthrough system and the coupling module.

5.3 Dynamic Simulation Procedure

Fountains and swales create or destroy dynamic objects parallel in the walkthrough system and in the simulator. If objects enter an assembly-line these movements in the simulation will be calculated in the walkthrough system. The switch arranges incoming parts due to their condition. If the switch is picked in the walkthrough system ("object-picking"), the system generates a message for the switch. Consequently, the switch changes its condition and arranges the parts in another way.

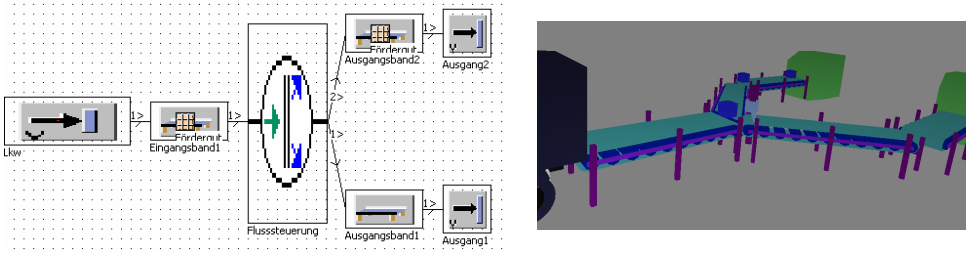

Figure 9: In the simulation all packages are running over the belt above (left).

Figure 8 shows the situation during the simulation in the EM-Plant and the walkthrough system. The packages are taken from the lorry and are then running over the belt above into the upper container.

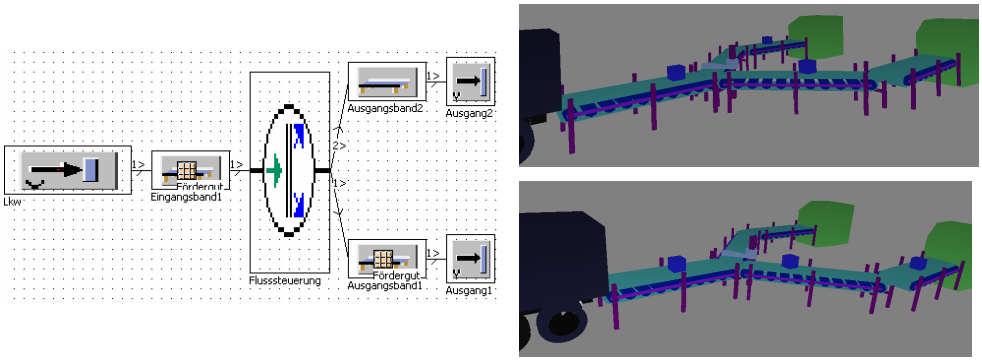

Figure 10: Now all packages are running over the lower belt (right). Picture above: conveyor belt directly after the switching in the walkthrough system by the viewer.

Figure 10 shows the simulation of EM-Plant and the walkthrough system directly after the intervention in the walkthrough system by the viewer: The viewer has clicked on the switches in the walkthrough system which has resulted in a conversion. Now all packages are running over the lower belt. The picture above just shows the disappearance of the last package on the upper conveyor belt into the container.
As oncoming events are not transmitted to the walkthrough system, no additional messages about time synchronisation have to be send. It is adequate, if the walkthrough system time does not outrun those events which have been transmitted last. Already small example-models transmit enough messages for a “smooth” coupling, which implicitly transmit a new time barrier for the walkthrough system. In the experiments our approach of the time synchronisation problem (constant follow-up time) works good. We estimate the follow-up time t_c by the communication time of the first transmission of a dynamic object. The experiments show that a estimated follow-up time t_c of some 100 milliseconds is enough for our tests. We made experiments with a reserve time t_o up to 1 second. But this does not improve the real time interaction in the walkthrough system.

6 Conclusion and Outlook

This paper presents chances and problems which may occur during a bi-directional coupling of simulators and walkthrough systems. We developed an architecture which presumes a stringent separation of simulators and walkthrough system. On this prerequisite we generated a method for the modelling and the description of a simple protocol. The time synchronisation problem is analysed thoroughly and a flexible solution is applied. A commercial manufacturing simulation tool was coupled to the walkthrough system in a bi-directional way. For this purpose, the simulation system was upgraded with modules, which implicitly contain the coupling. These modules are able to create static and dynamic objects in the walkthrough system and can therefore generate a 3-dimensional world. Dynamic objects can be moved and erased from out of the

simulation via the protocol. The walkthrough system is able to take in user interactions, attribute those to the objects, send them to the simulator and trigger actions there.
The following paragraphs contain considerations about further developments in the field of such implementations. They offer potential for other theses:

6.1 More Flexible Interfaces in the Protocol through XML

In the current realisation data is transmitted via a simple proprietary protocol. This is sufficient for a validation of the method. Is the system meant to be employed in the long run, though, and should transmitted functionalities be upgraded, such a system will turn out to be unstable and difficult to be overhauled. The protocol is not efficiently defined if another simulator or another walkthrough system has to be employed. The creation of a grammar which is based on XML would be a possible solution. Especially as that would ensure flexibility through simple addition and easy reading. Neither simulators nor the walkthrough system which we currently employ already contain XML Parser.

6.2 New Directions for Rendering Algorithms

Many traditional culling and approximation rendering algorithms for highly complex scenes try to reduce the scene complexity by removing triangles of objects which are not in the focus of the visitor. The problem is to learn what is in the "focus" of the visitor. If no special focus is known, the walkthrough system assumes that the focus is given by the position and viewing direction of the visitor. The bi-directional coupling of the simulation system and walkthrough system offers more than this simple information and therefore leads to new efficient rendering algorithms for simulation systems:
Important Objects in the focus of the visitor: The simulation is able to give information about important objects of the scene, e.g. if the visitor is looking to a machine in a large factory hall. In this situation the simulation can demand a very high model quality of the machine and a low quality for all other objects around the machine, because in this situation the machine is the most important object for the visitor. If the machine is a punching machine for car doors, then the simulation can demand e.g. high quality for the car door 3D-model and center of the punching machine, medium quality for the punching machine 3D-model, and low quality for all other 3D-models of the factory hall. As the simulation system knows exactly what kind of simulation objects are important, we can use rendering algorithms which allow different levels of details (LOD) of nearby objects. In contrast, traditional algorithms use high quality for nearby objects and low quality for far away objects.
Prefetching of rendering algorithms: In contrast to simulation systems traditional walkthrough systems only have few information of future scene situations available. In many situations the simulation is able to look in the future, e.g. if a truck arrives in a factory hall and some kind of fork-lift truck will move to the truck and repack the cargo. The movement of the fork-lift trucks is known in advance, because of the arrival of the truck. This is an important information for occlusion-culling and approximation algorithms. They can prefetch approximations (e.g. from hard disc) and culling computations for the fork-lift truck. Therefore the rendering algorithms can act according to future real-time requirements of suddenly coming scene objects in advance.

Limited repeated life cycle of objects: Some objects of the simulation have limited lifecycle, e.g. packets of a conveyor belt. E.g. let O_j be an Object generated at a fixed time t_i and die in the simulation at time t_{i+1} . This object O_j will be generated at time t_j again and die at time t_{j+1} . We assume that the object O_j will have the same 3D-model and the same path through the scene at time t_i and t_j . Therefore we can use exactly the same approximations (e.g. dynamic computed textures etc.) for the two lifecycles and speed up rendering times. This kind of information has to be transmitted to the walkthrough system by the simulation system.

6.3 Potential for Simulations

A bi-directional coupling with a walkthrough system enables a simulation system to register the position, the movements and the interests of the viewer. This provides potential for procedures that simulate exactly those objects which are in the focus of the user. It furthermore offers the opportunity to simulate parts of the simulation, which are farther away and more coarse, while using less computer-time. Does this resolution continuously follow the movements of the viewer, he will get the impression of a complete and detailed simulation, though it has not been calculated.

Acknowledgment: We would like to thank Lydia Vonderheide and Vitali Ginter for implementing the coupling module, modelling 3D-representatives and EM-Plant.

References

[Abr00] Abrash, M.: *Ramblings in Realtime*, Homepage, http://www.bluesnews.com/abrash, 2000

[Abr97] Abrash, M.: *Graphics Programming Black Book*, The Coriolis Group, Inc., 1997, also available: Homepage Dr.Dobb's Journal, http://www.ddj.com, 2001

[DMS99] Defense Modelling and Simulation Office (DMSO): High Level Architecture Homepage, http://hla.dmso.mil, 1999

[Jes01] Jessen, U.: *Konzeptioneller Ansatz für die bidirektionale Kopplung von ereignisorientierten Simulationswerkzeugen und Virtual Reality Systemen*, In: K. Panreck u. F. Dörrscheid: Frontiers in Simulation Band: 8: Simulationstechnik, 15. Symposium in Paderborn, 2001

[KlG99] Klingstam, P.; Gullander, P.: *Overview of simulation tools for computer aided production engineering*, In: Computers in Industry, Nr. 38 S. 173-186, 1999

[LSU99] Lantzsch, G.; Straßburger, S.; Christoph Urban: *HLA-basierte Kopplung der Simulationssysteme Simplex III und SLX* In: O. Deussen, V. Hinz, P. Lorenz: Simulation und Visualisierung'99, Magdeburg, 1999

[Swe99] Sweeney, T.: *Unreal Networking Architecture*, Homepage, http://unreal.epicgames.com/Network.htm, Epic MegaGames, Inc., 1999