Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

MULTI-USER SUPPORT AND MOTION PLANNING OF HUMANS AND HUMANS DRIVEN VEHICLES IN INTERACTIVE 3D MATERIAL FLOW SIMULATIONS

Matthias Fischer
Bengt Mueck
Kiran Mahajan
Michael Kortenjan
Christoph Laroque
Wilhelm Dangelmaier

Heinz Nixdorf Institute
University of Paderborn
Fürstenallee 11
33102 Paderborn, GERMANY

ABSTRACT

The visualization of simulated production processes is used for their analysis. Huge plants are normally planned by a team. So a solution for many users who are modeling and interacting with a running model in an immersive 3D environment is required. We discuss an approach where several users work cooperatively on one simulation model. To optimize their work, the users need some guidance. For this we suggest small maps and arrows to guide the user to significant objects (machines). In many production scenarios, objects (forklifts, workers) are moving in an unguided fashion. In actual implementations these paths have to be modeled manually. In spite of taking these efforts, we are presenting an automated approach which is based on the 3D layout of the plant. If the user as part of the simulation is standing in the way of the object, the object stops in our approach (as hopefully in reality).

1 INTRODUCTION

Simulation and visualization are well-known methods for the understanding, analysis and discussion of manufacturing processes. It is much easier to communicate a workflow with a working virtual reality implementation than just looking at marks moved around (like Petri-Nets) on cryptic symbols (Lawrence 2003).

Huge systems are planned by a team of several users working on one system. Every member of the team is sitting at his own workstation moving through the 3D environment of the simulation in order to inspect and analyse the simulation. Every member gets his own mind and ideas walking his own way through the 3D environment. However, once in a while, users meet in the 3D environment to discuss problems of the simulation. Synchronisation, interaction and communication is needed. Every user should see the actions of the other users. If somebody has a problem, he must be able to communicate his problem.

In actual visualizations of manufacturing processes, the viewer can move around freely and unguided (Mueck et al. 2003). If the system is huge, the user can lose his orientation. He needs some help to orientate and especially to find the next point he wants to reach.

Even if he knows where he is and where he wants to go, he is not really part of the system. In traditional simulation systems, he is mostly a *passive viewer*. For many applications this might be enough, but if he wants to interact with the simulation, he needs to be a part of it. And vice versa, the simulation needs to interact with him. Some things, like movements of parts (Bluemel et al. 2003), are still available for this. But a lot of things are still missing. Especially the interaction with workers and forklifts, which are travelling around, is needed. Typically in actual implementations, forklifts do not take notice of the user and the avatars of the workers are moved like the other moving objects. They do not even take account of the actual layout of the plant. But in reality a moving object, e.g., forklift, should stop if a visitor or even a group of visitors is standing in his way. And the forklift also should change its path if the layout of the plant has changed. Workers are part of the simulation. This article will describe approaches and the experiences from implementing and experimenting with these approaches for all the problems mentioned.

2 STATE OF THE ART

We address two problems with our system: first, the cooperative work of a team that works jointly on a simulation

model using a simulator that supports a 3D rendering of the simulated model. Second, the automatic computation of paths for unguided objects such as forklifts.

2.1 Working in Teams

Today integrated packages like AutoMOD (Rohrer 2003), eM-Plant or Taylor ED (Nordgren 2001) are used for material flow simulations (Klingstam and Gullander 1999). With these tools, the user is able to model and execute models of production processes in one simulation environment. He can also analyze his model in a 3D environment. The whole process of modeling, execution, analysis and modification takes place on one person's computer. If more than one user wants to edit a model, the model has to be split. Each user works on his specific area and the model can be composed again from these parts. In this case, each modeler has only a local view of the model during the modeling process. This is a very rudimental way of cooperative working. If the complete model is to be analyzed, a simulation has to be carried out. Only one camera position is possible during one run. So the whole team has to use the same perspective or use different runs. Cooperative working during the simulation run is not supported.

2.2 Moving Objects

For the modeling of moving objects, different ways of modeling are possible. Assume that packets enter at point A and are carried by the forklift to point B where they are loaded and processed. Because of the modeling connections between point A and B, in *Taylor ED,* the forklift is using the direct path to travel between these two points. If the layout is changed, the forklift does not automatically change its path to suit the new layout. In *AutoMOD*, the motion of an element between two points is to be planned by the modeler using "non-flexible" paths. If the model of the factory consists of many machines and many alternative paths for several moving elements, the approach used by Taylor ED would certainly not work (to automatically determine the motion path), whilst (both TaylorED and AutoMOD) also fail to automatically change the motion path of the moving object based on a new layout by the modeler to avoid collisions with other elements.

3 CONCEPTUAL OVERVIEW

The goal of our system is the joint work of multiple users in a 3D simulation environment. The users control the simulation in a virtual 3D environment of the simulated system. They model and parameterize the system in a 3D environment. The results of the simulation are visualized online during runtime of the simulation. During the simulation they change parameters by interacting with the 3D scene and observe the effect on the simulation model. To implement this, the 3D rendering workstation and simulator are coupled bidirectionally. The system has the following advantages:

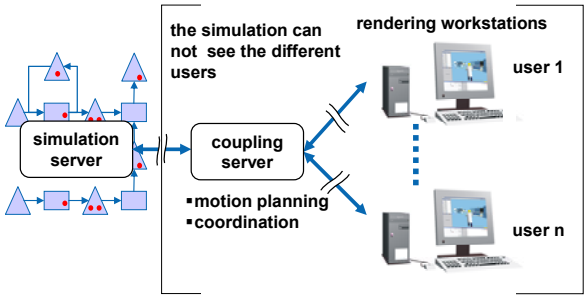

Figure 1: Architecture of Our System Consisting of Distinct Workstations for Simulation and Rendering.

1. Intuitive observation of the simulation at runtime.
2. Common modeling, analyses, and discussion of simulation experts.
3. Intuitive presentation of simulation results for non simulation experts.

Our system consists of a simulation server, a coupling server, and an arbitrary number of rendering workstations (see Figure 1). The material flow simulation runs on the simulation server. Each user has its own rendering workstation. He uses the rendering workstation to control the simulator and to observe the simulation process rendered as 3D virtual scene. The coupling server connects all rendering workstations with each other, and performs specialized tasks, e.g., computation of motion planning, communication and other tasks that should not be executed by the simulation server.

It is important to execute the simulation and the rendering of the simulated scene on distinct servers: first, we avoid a performance problem. Real-time rendering of a virtual scene needs the whole performance of a graphics workstation. In the same way, complex online simulations need the whole performance of the server (Mueck et al. 2002). Second, for the joint work of multiple users, each user needs his own workstation to analyze and to observe the simulation model independently from other users.

2D/3D Graphical User Interface: Figure 2 shows the overall view of the graphical user interface (GUI): the upper part shows the 3D rendering of the simulated scene. Beside the 3D rendering is a list of significant objects and processes. Below the list is a property panel. In the lower part we have a chat panel and a list of users who are logged in. Beside the chat panel is a minimap that renders a 2D view of the 3D scene. The 2D/3D graphical user interface should fulfill three main functions:

1. The user should observe and analyze the simulation run (Section 6).

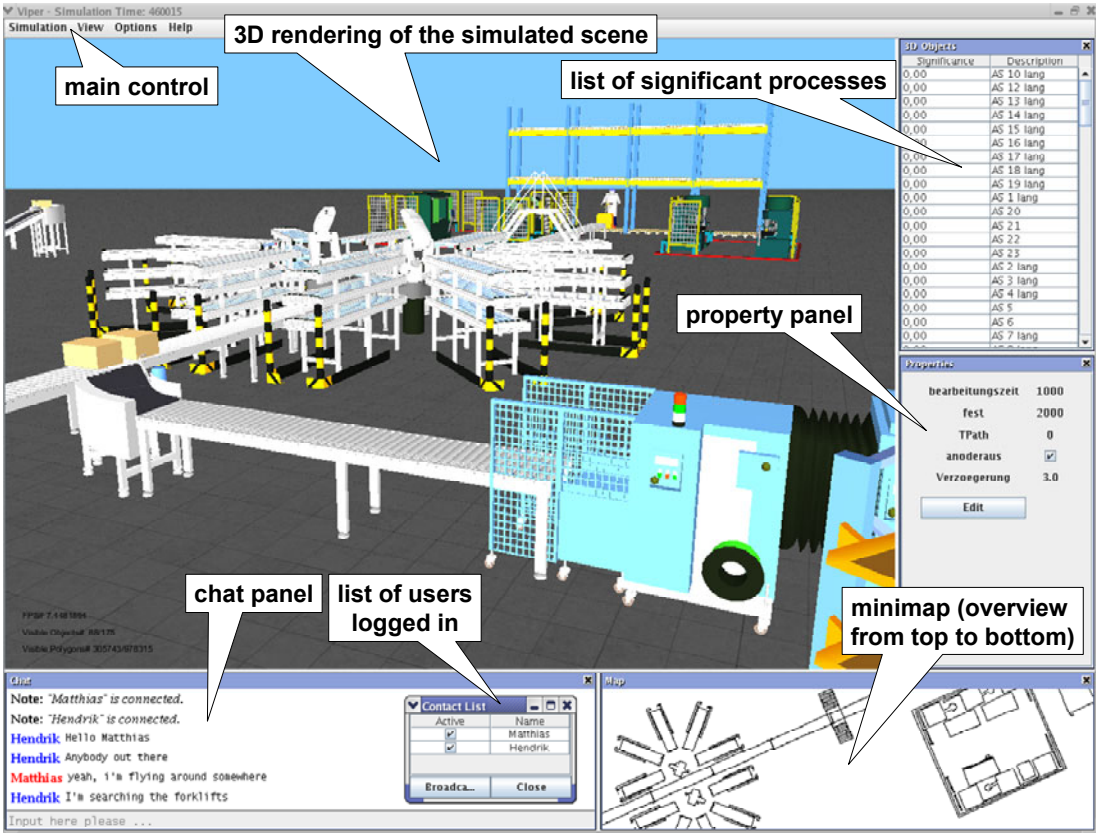

Figure 2: The Graphical User Interface of the 3D Rendering Control.

2. Multiple users meet in the scene, communicate and discuss with each other (Section 4).
3. The user interacts with the simulation in the 3D scene and modifies simulation parameters. The simulation should respond because the user is an *active viewer* (Section 5).

4 MULTI-USER COMMUNICATION AND VISUALIZATION

In a multi-user system, users meet in the scene, discuss problems, and analyze the simulation. Therefore, the users must have the ability to communicate with each other. Also the visualization of other users is important in order to point to machines and other things.

4.1 User Communication

The system supports the communication between distinct users who are connected with the simulator. For this they use a chat panel as shown in Figure 2. They write a message in an input line, which is sent to all connected users. The chat panel displays all messages that are sent by users. Right beside the chat panel, the user sees a list of users who are connected with the simulator.

4.2 Visualization of Users

Our implemented system visualizes the users of the scene with avatars. As shown in the example of Figure 3, the avatar is a simply modeled person. The avatar is positioned in center of the lower border of the image and moves when the user moves to another position. To prevent occlusion, the avatar can be switched off. The avatars of other users are placed at their current positions in the scene. See the woman with the red jacket (user 2) in Figure 3. The avatar has the name of the user on his head.

We notice the movement of other users by the movement of their avatars. It is important that we make out the viewing direction of other users, e.g., in situations where two users discuss a process and point to a machine.

5 MODIFICATION AND INTERACTION WITH THE SIMULATION

Our system supports the interactive modification of parameters during the runtime of the simulation. Many parameters are changed by a property panel which seems to be a good solution. However, an *active viewer* is the more natural option to interact with the scene especially if the scene consists of moving objects. The user is part of the scene and interacts with the objects such as forklifts.

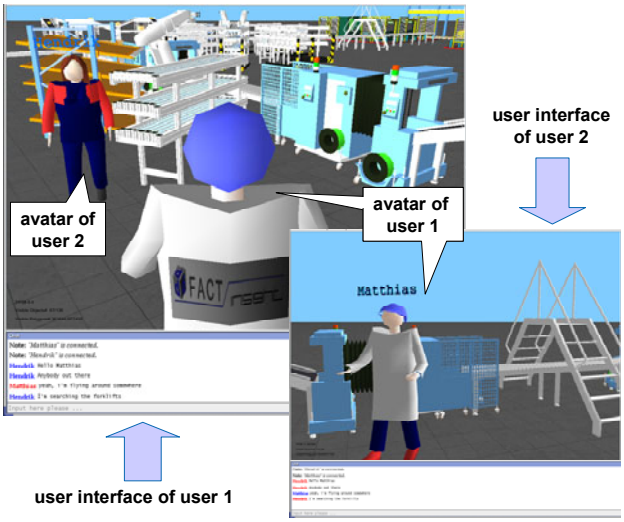

Figure 3: Two Users, Each Represented by a 3D Avatar, Analyze a Significant Process (User 2 has Switched Off his Avatar).

5.1 Modification of Model Parameters

The user can change the parameters of the simulation during the runtime of the simulation. He picks a machine in the 3D scene to do this. After picking the machine, the property panel (see Figure 2) shows the parameters of the machine. The user can press an edit button to change the parameters. After editing, the simulation run continues and the user observes the influence of his changes immediately.

5.2 Interaction of Active Viewers with the Simulation

The goal of the system is an intuitive observation of the simulation at runtime. The user should observe how packets are processed and forklifts are doing their tasks. The first image in Figure 4 shows how packets move across the conveyor belts. Forklifts transport packets between processing points. The user should see how forklifts and packets move. Figure 4 shows how a forklift arrives at a conveyor belt and picks up a packet. Afterwards, the forklift transports the packets to the destination. The forklift arrives at the destination and unloads the packet on the conveyor belt. The user can observe the whole process in the simulated 3D scene.

However, interaction with the simulation should also be possible. It should be possible to generate unexpected events. For example, it should be possible to stop a forklift. The fifth image in Figure 4 shows how a user moves in front of a moving forklift. The forklift recognizes that the user is a barrier and stops. Only when the user moves away from the path, can the forklift drive on and move to the destination. This example shows that the user can interact with the scene not only using 2D property panels, but also through navigation and movement in the scene. The user is an *active part* of the scene and not a *passive viewer*.

6 ORIENTATION AND NAVIGATION

For efficient usage of the 3D visualization of the simulated scene, the 3D system must support easy orientation and navigation in the scene. Furthermore, the system must visualize and highlight *significant objects* and processes so that the user can easily detect them. In the following sub-

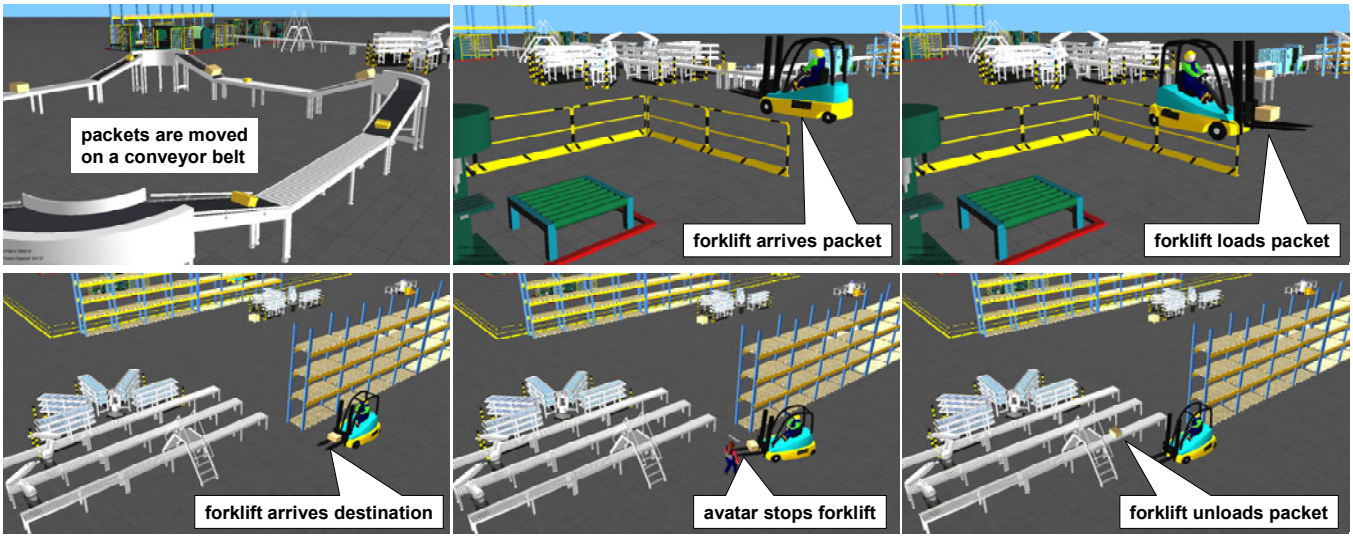

Figure 4: Packets are Transported on a Conveyor Belt and a Forklift Moves to a Conveyor Belt, Picks Up a Packet, Transports the Packet to the Destination; at the Destination the Forklift is Stopped by a User, Waits and Unloads the Packet After the User Moves Away.

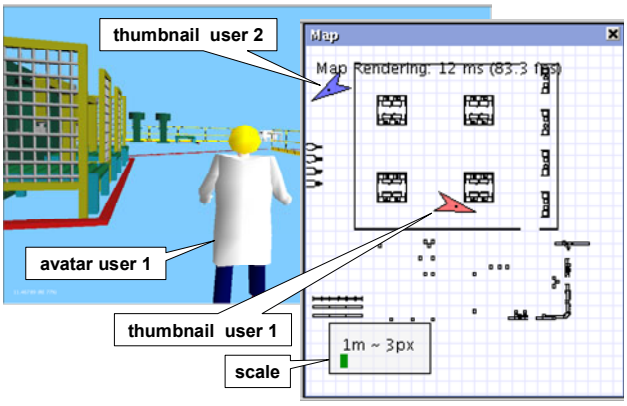

Figure 5: Because of the Thumbnail of User 2, User 1 Knows the Position of User 2 Who is Occluded by Machines, Thus the Minimap Helps User 1 to Orientate.

section we describe how our implemented system supports these demands.

6.1 Disorientation and Overview

A problem of 3D environments is orientation in the virtual world. Especially in environments of material flow processes, e.g., in a large factory hall, the area is large and often changes during the planning process so the user can lose his way and orientation. To prevent disorientation, the user must concentrate on finding the right way to the desired destination. However, the purpose of the 3D visualization is to observe and analyze processes and not to waste time with finding and exploring a way through the scene. Therefore, the system must make orientation as easy as possible.

Our system supports easy orientation through the implementation of a *minimap* that shows an overview of the scene. Figure 5 shows the 3D scene on the left side and the corresponding minimap on the right side. The overview of the minimap consists of a two-dimensional floor plan of the scene, e.g., the factory floor. We automatically draw the skeletal outlines (silhouettes) of all static objects such as machines and conveyor belts, and of all moving objects such as forklifts and users. The outlines of the 2D view are computed automatically from the 3D model, we use the same outlines as for motion planning (see Section 7). The users are drawn as thumbnails in the minimap. Therefore, a user can easily find a way to neighboring users who are occluded by large machines or other objects. Our minimap supports simple:

- Finding a way to the destination,
- Exploration of the factory and environment,
- Finding of neighboring users.

Currently we use arrows, which look like compass needles, as thumbnails for the users. The compass needles

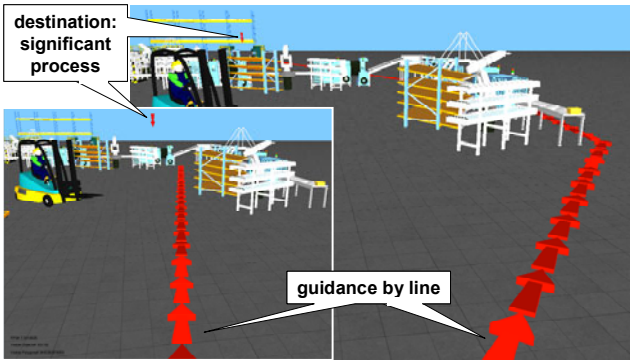

Figure 6: Lines on the Ground Show the Way to the Significant Process.

show the line of sight of the user. This supports easier orientation in the scene.

6.2 Guidance of Users

In traditional 3D simulation environments, the user must search and find the way to a significant process himself. The way he walks is based on experience and some expectations in the best case. But in many cases his path is also influenced by luck. Instead of observing and analyzing the processes around him, he must concentrate on finding the way to the significant processes. This is unstructured and inefficient. The users are losing time. In our approach the user should be guided to the significant processes.

To avoid disorientation and confusion, the user must be guided along regular traffic routes. A movement "through" all machines directly to the destination, which is possible in a virtual 3D environment, would confuse the user. Therefore, our system computes a path from the current position of the user to the machines with the significant processes. The path between the significant processes and the user is shown with lines on the ground (as suggested by Darken and Peterson (Darken and Peterson 2002)). If the user wants to go to a significant process, he only has to follow these lines. The system also checks if the user collides with moving objects such as forklifts. Figure 6 shows an example. In our current implementation, the line consists of a sequence of red 3D arrows. The arrows appear if the system detects a significant process, and disappear when the user reaches the machine with the significant process.

6.3 Visualization of Significant Objects and Processes

In our approach we interact with the underlying simulated system on the basis of *significant processes*. A significant process is a process which will recover the system from disturbances. We consider probabilistic disturbances, e.g., breakdown of machines and arrival/cancellation of orders. If the user wants to improve the performance of the production

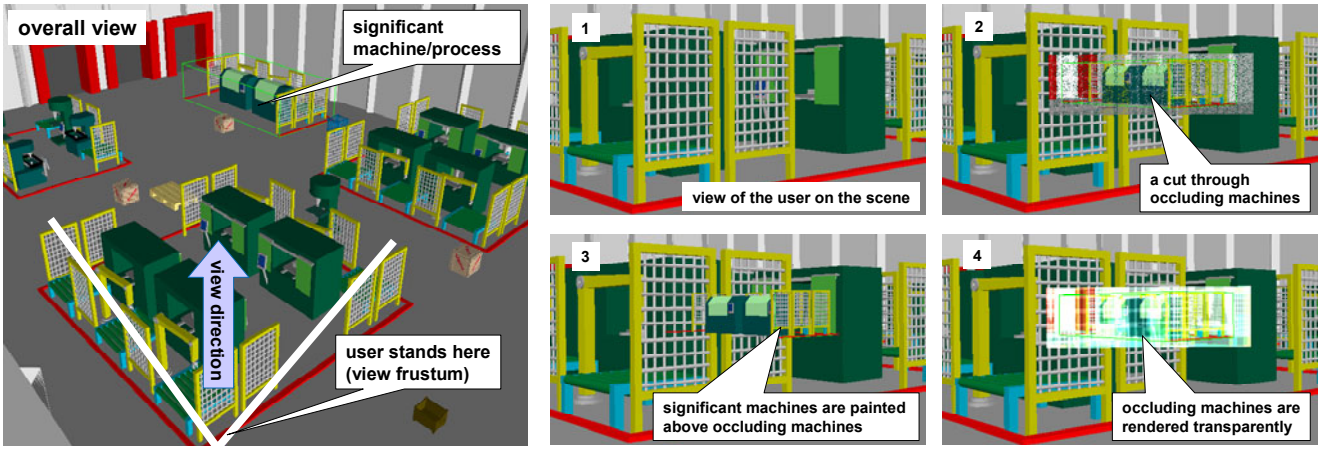

Figure 7: The User Stands in Front of a Machine that Occludes a Significant Machine; to Render Significant Objects Image 1 Shows the View of the Occluded Machine, Image 2 Cuts Through All Occluding Machines, Image 3 Paints Significant Machines Above Occluding Machines, and Image 4 Renders Occluding Machines Transparently.

system, the significant processes are the processes he should take care of. The system must detect these processes automatically and help the user solve potential problems.

Thus they are important for the user to watch. But typically the user is not standing direct in front of these processes. The user has at least to be notified where significant processes occur. Therefore, the user must walk through the production hall. He cannot see very far, because he is surrounded by machines. See Figure 7 for example; the left image shows an overall view of the scene. The significant machine stand at the end of the hall. The user stands in front of machines that occlude the significant machine. Image 1 of Figure 7 shows the view of the viewer onto the scene.

6.3.1 Rendering of Occluded Significant Objects

Because significant machines are important for the user of the system, we tested three techniques to render significant machines: one possibility is to render the significant machines above the occluded machines (see Image 3 of Figure 7). This solution ensures that the user sees always the closest significant machine. These solution is usable if many machines stand between the occluding machine and the significant machine. However, sometimes it is difficult to distinguish the occluding machine from the significant machine (shape). The second approach renders all objects between the occluding object and the significant machine transparently (see Image 4 of Figure 7). Therefore, the user sees the occluded machines as well as the significant machines. This solution works well if only a few objects stand between the occluded object and the significant object. Our third approach cuts a tube through the occluding objects so that the user sees the significant objects behind them (see Image 2 of Figure 7).

6.3.2 Rendering Significant Objects with Different Significance Level

If the user is guided through the scene to a significant object, he looks around and observes other significant objects and processes, thereby analyzing the processes. Therefore, it is important to give him as much information as possible about the processes without confusing him. We marked the significance level of objects using several background colors. See Figure 8 for an example: the occluded machine right has a high significance level that we mark with a green background. The occluded object left has a low significance level that we mark with no extra background color.

6.3.3 Highlighting Significant Objects by Arrows

One possibility to inform the user is to show him a text list of significant points and machines (see the list of significant processes in Figure 2). This would be a time consuming solution because the user must permanently read the list in or-

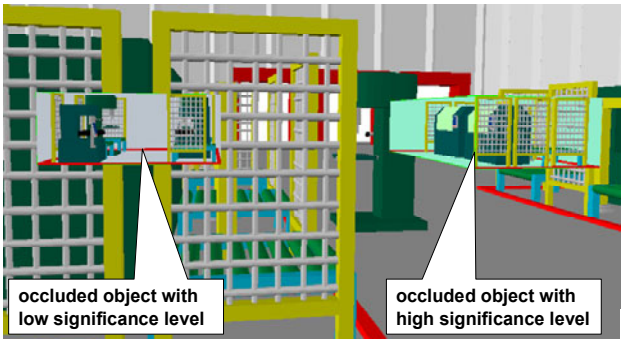

Figure 8: Occluded Objects with High and Low Significance Level.

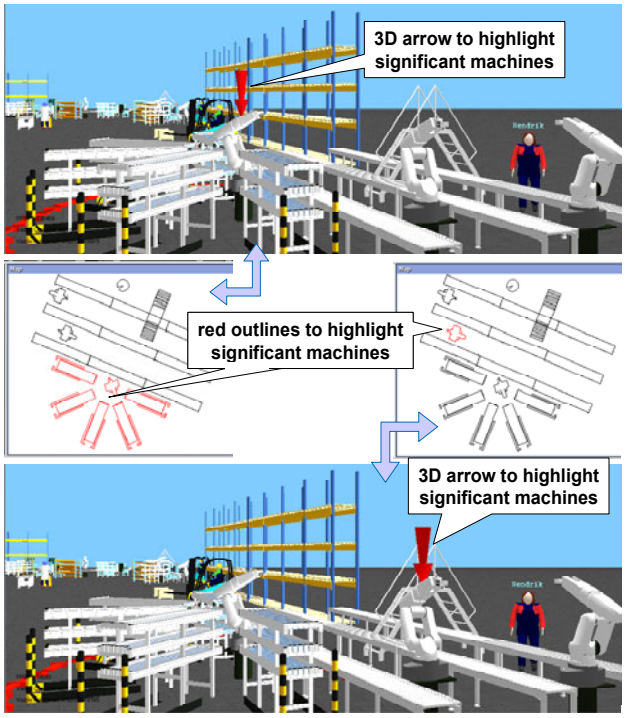

Figure 9: The Virtual Scene Renders Red 3D Arrows and the Minimap Shows Red Outlines to Highlight the Significant Machines.

der to analyze which of the significant objects are important for him. We speed up this process with a visual solution. We highlight significant objects in the scene using red 3D arrows that are placed at the top of a significant object (see Figure 9 for two examples). Thus the user can easily get a visual overview of which machines are significant and which not. The visual detection of significant machines works faster than the textual detection. The minimap renders significant objects using red outlines. So the user can easily detect which parts of the object belong to the significant machine. This is simpler than only a single 3D arrow.

7 MOTION OF HUMANS AND HUMANS DRIVEN VEHICLES

The user has to model and parameterize the simulation and finally view the simulation in a virtual environment. After analyzing the system, the user might wish to carry out changes in the layout, parameters, etc. of the simulation model, which also includes determining new motion paths for objects such as forklifts and automated guided vehicles. A motion planning algorithm is necessary that automatically determines the paths for moving objects depending on the new model layout without colliding with other objects of the virtual factory.

7.1 Problems to Solve

The problem is to automatically find motion paths for moving elements based on the layout changes done by the modeler. This should happen without colliding with other "obstacles" in the factory. The system must unburden the modeler from modeling the motion paths of the vehicles. We distinguish between the motion of *autonomous vehicles* and the motion of *constrained objects*.

Autonomous vehicles are objects such as humans and human driven forklifts. The modeler only defines the start and end position. The paths must be computed automatically by the system. Constrained objects are objects such as packets that move over a conveyer belt. The modeler defines the path of the object by choosing the conveyor belt.

Further problems arise from the interaction with avatars and collisions of autonomous vehicles. The system must stop a forklift if a user stands in the path of the forklift. Furthermore, the system must prevent all collisions of the autonomous vehicles.

7.2 Our Approach

Our implemented system supports the motion of autonomous and constrained vehicles. The motion of constrained vehicles, such as packets, is controlled by *token paths*, which are defined by the modeler. The token paths consist of a sequence of positions. The system moves the objects along the path interpolating between two positions on the token path.

The motion of autonomous vehicles is computed automatically by the system. The motion planning algorithm consists of three steps :

1. Create 2D outlines from the 3D models of objects and obstacles of the factory.
2. Create a scene specific graph to represent possible motion paths for the movable object.
3. Search for a path and collision prevention.

In the preprocessing and after changes in the layout of the scene, an object specific phase is performed to create a 2-dimensional outline from a 3-dimensional model. This outline is also used for the rendering of machines in the minimap. The second phase creates a scene specific graph representing possible paths for the moving objects. During runtime, this graph is used to search for a path from the current position of a moving object to its destination and for the guidance of users (see Section 6.2). We describe the single steps of our approach in the next subsections.

7.2.1 Outline Generation

Models of machines and moving objects are given as 3-dimensional objects, which are created with 3D modeling

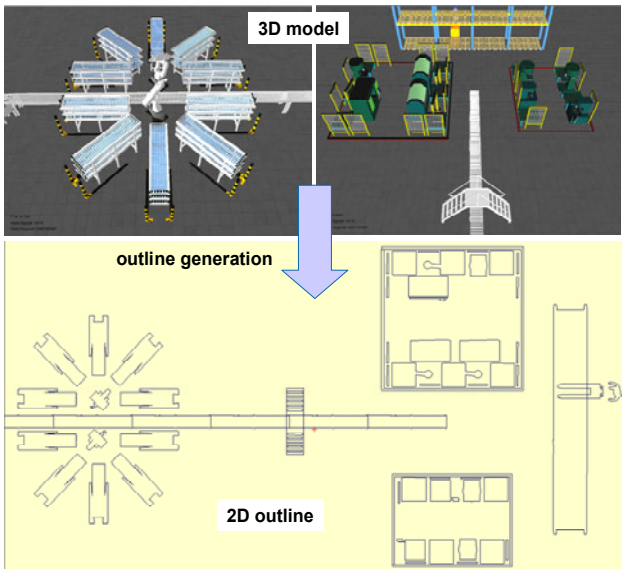

Figure 10: The 3D Model of Three Machine Cells and a Forklift, and the Corresponding 2D Outline.

programs such as 3d Studio Max. Motion planning takes place on a plane as all vehicles move on the factory floor. Thus we need a 2-dimensional representation of our models. We implemented an algorithm that computes the outline of the model's orthogonal projection from the 3D model. The example in Figure 10 shows the 3D model of two machine cells, several conveyor belts, and the automatically generated outline.

Simply projecting the 3D model does not take into consideration that objects might be of different widths. The object in Figure 11 is in a feasible position that overlaps the obstacles highest outline (highest dashed line). To circumvent this, we discretize the model's height into a fixed number of levels. We use the outline of the height next larger than the object's size. Since we always want to have a secure distance from the object to the obstacle, we are not interested in the outline of obstacles clipped immediately above the objects height. Therefore, we use the smallest level which is high enough to maintain a safe distance.

7.2.2 Generation of Motion Paths

For each type of moving object we create a motion graph which stores the position and direction of view of the moving object on its edges and nodes. In order to achieve this, the motion graph is constructed in three steps: construction of a extended outline, construction of a motion graph describing a directed path around each obstacle and between the obstacles, and mapping docking points to the graph nodes identifying possible connection points.

Step 1: A scene is specified by the obstacles, e.g., the machine cell in Figure 12. The union of the obstacles' outlines determines regions impassable for moving objects,

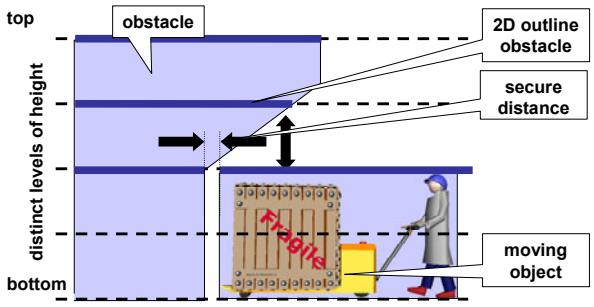

Figure 11: We Generate the Outline of the Objects for Distinct Heights of the Objects.

e.g., see the outline of machine cells in the left image of Figure 13. For each kind of moving object, we construct a motion graph representing potential paths in this scene. First, we choose the outlines of the obstacles depending on the object's height. The object's dimensions characterize a minimal distance to the obstacles. Each object has an orientation axis. If the object moves, it has to be rotated such that the axis points along its direction of movement. Each object has a reference point. We move this point along every obstacles outline, keeping a distance such that the object never collides with the obstacle if its orientation axis is aligned in the direction of the edge. This results in a new enlarged outline that is shown in Figure 12.

Step 2: The extended outlines define the motion graph (middle image of Figure 13). The endpoints of the extended outlines result in nodes and an edge is inserted between its endpoints for every line. These edges represent a path around an obstacle. Every node of the motion graph stores its position, the orientation of the moving object and the geometry used for collision prevention. Finally, we compute connections between neighboring obstacles and insert edges to the graph. These edges are used to cross free space between the obstacles. Since we are interested in finding shortest paths, the edges are weighted with their length. The right image of Figure 13 shows the motion graph with the motion blur (green) of the geometry of a moving object (forklift). The geometry of the moving object is used for collision prevention.

Step 3: Our database additionally contains so called *docking points* which can be defined as positions where objects can interact with obstacles. A forklift must not drop its load anywhere near its target machine, but at a certain position. These docking points belong to the obstacles. The simulation kernel will order an object to move to a specific docking point. In the outline of Figure 12, additional nodes are inserted which represent these docking points. These are special target nodes for the moving objects. The left image of Figure 13 shows the outlines of the machines and the inserted docking points (green) for the forklifts (red rectangles).

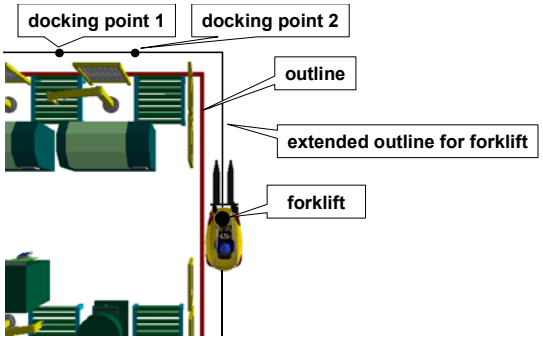

Figure 12: The Outline of a Machine Cell and the Extended Outline Used for a Forklift to Move Around the Machine Cell.

7.2.3 Moving Objects Through the Scene

At runtime, we use the motion graph to find a path from an object's current position to a given goal point. We move objects among the edges of the motion graph crossing the nodes of the graph. We use a shortest path algorithm to find the shortest path from the source to the destination. The obtained path is used for collision prevention. We have an optimistic approach: since our graph implies a right handed traffic system, we expect collisions to rarely happen. We only predict the future for a small time window in which collision is solved by waiting. At this point we use the paths to compute the positions of the moving objects in the plane. If simulation time has moved beyond the end of our time window, we compute the next time window.

Combined with our visualization system another kind of collision can occur than two objects crossing paths. Since we want as much immersion of the user as possible, the environment should react to him. If the user blocks the movement of an object (he stands on the path), the object has to stop. We cannot control the user or estimate for how long he will stay there, so we just wait until the user leaves the path instead of trying to get around him (*active user*).

8 CONCLUSIONS

The visualization of simulations of production processes is a common method used to analyze and discuss variants in a team. We showed an approach where all team members can explore the simulation model at the same time. Interaction caused by one member takes effect in all visualizations of the simulation run. So all users can make their individual observations at the same time on the same model. If the process is huge, users need support to effectively find their way to their specific destination in the virtual world. We support the orientation and navigation of the team members with a small map, automatically generated from the 3D model of the simulation. Additionally, our system shows arrows on the way to and on top of the important objects. So the user has the option to be directly guided to his point of interest instead of being potentially confused. In real production plants, forklifts and humans typically move wherever they have space. They are not guided. And they need space for maneuvering. As the transport time has an impact on the simulation results, not mentioning the real paths leads to inaccurate results. In traditional simulation systems the traveling times are estimated and the forklifts move directly between the departure and the destination or on specially guided paths. As our system supports the automatic generation of outlines of the 3D models used, we use this for a much more accurate calculation of the these movements, which is based on the layout given by the 3D model. If the layout changes, the paths of the forklifts will also be changed. For good immersion of the user, our approach also stops forklifts (as the forklift driver should do in reality) if the watching user is standing in its way and would like to stop the forklift.

ACKNOWLEDGMENTS

This work is partially supported by DFG grant DA155/29-1 "Benutzerunterstützte Analyse von Materialflusssimulationen in virtuellen Umgebungen"

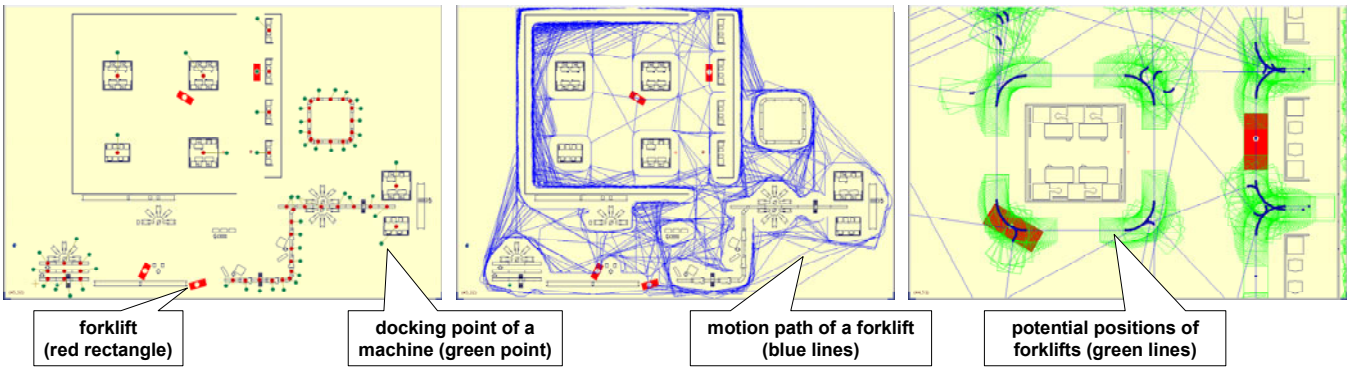

Figure 13: The Outline of Machine Cells, Forklifts and Docking Points are Shown in the Left Image, the Computed Motion Graph in the Middle, and the Motion Graph with Potential Positions (Motion Blur) of the Forklifts in the Right Image.

(BAMSI). Many thanks to the students of the project group "Wege und Bewegungen in virtuellen Produktionsumgebungen" (wubivipu) for implementing parts of the system.

REFERENCES

Bluemel, E., Hintze, A., Schulz, T., Schumann, M. and Stuering, S. 2003. Virtual environments for the training of maintenance and service tasks, In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 2001-2007.

Darken, R. P. and Peterson, B. 2002. Spatial orientation, wayfinding, and representation, In *Handbook of virtual environments: design, implementation, and applications*, ed. K. M. Stanney, 493-518. New Jersey: Lawrence Erlbaum Associates.

Klingstam, P. and Gullander, P. 1999. Overview of simulation tools for computer aided production engineering, In *Computers in Industry*, 38: 173-186.

Lawrence, P. 2003. Visual simulation in manufacturing management: some observations, In *Industrial Simulation 2003 – 1st International Industrial Simulation Conference*, ed. J. C. Guerri, A. Pajares, and C. Palau, 331-335.

Mueck, B., Dangelmaier, W., Fischer, M. and Klemisch, W. 2002. Bi-directional coupling of simulation tools with a walkthrough system, In: *Simulation und Visualisierung*, ed. T. Schulz, S. Schlechtweg, V. Hinz, 71-84.

Mueck, B., Dangelmaier, W. and Fischer, M. 2003. Components for the active support of the analysis of material flow simulations in a virtual environment, In *Proceedings of the 15th European Simulation Symposium*, ed. A. Verbraeck and V. Hlupic, 367-371.

Nordgren, W. B. 2001. Taylor enterprise dynamics. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 269-271.

Rohrer, M. W. 2003. Maximizing simulation ROI with AutoMod, In *Proceedings of the 2003 Winter Simulation Conference* ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 201-209.

AUTHOR BIOGRAPHIES

MATTHIAS FISCHER studied computer science at the University of Paderborn, Germany. Since 1995 he has been a research assistant at the Heinz Nixdorf Institute. In 2005, he received a Ph.D. for his work on distributed virtual environments. His research interests are computer graphics, real-time rendering algorithms, and distributed computing. His e-mail address is <mafi@upb.de> and his Web address is <http://www.upb.de/cs/mafi>.

BENGT MUECK studied computer science at the University of Paderborn, Germany. Since 1999, he has been a research assistant at the Heinz Nixdorf Institute. In 2004, he received a Ph.D. for his work on dynamic multiresolution modeling of huge simulation systems. His research interest is development of simulation tools for logistic systems. His e-mail address is <mueck@hni.upb.de> and his Web address is <http://wwwhni.upb.de/cim>.

KIRAN MAHAJAN studied mechanical engineering at the Delft University of Technology in The Netherlands with a specialization in production engineering. Since 2004, he is a research assistant at the Heinz Nixdorf Institute. He has a bachelor's degree from the University of Pune (COEP), India. His research interests are interactive analysis of manufacturing simulations and 3D visualization. His e-mail address is <kiran@hni.upb.de> and his Web address is <http://wwwhni.upb.de/cim>.

MICHAEL KORTENJAN studied mathematics at the University of Osnabrück, Germany. Since 2003 he has been a research assistant at the Heinz Nixdorf Institute. His research interests are algorithms for computer graphics and real-time rendering. His e-mail address is <mkortenj@upb.de> and his Web address is <http://www.upb.de/cs/mkortenj>.

CHRISTOPH LAROQUE studied business computing at the University of Paderborn, Germany. Since 2003 he has been a Ph.D. student at the graduate school of dynamic intelligent systems and research assistant in the group of Prof. Dangelmaier, Business Computing, esp. CIM. He is mainly interested in material flow simulation models and the "digital factory". His e-mail address is <laro@hni.upb.de> and his Web address is <http://wwwhni.upb.de/cim>.

WILHELM DANGELMAIER studied Mechanical Engineering at the University of Stuttgart, Germany. In 1981, he became director and head of the Department for Corporate Planning and Control at the Fraunhofer Institute for Manufacturing. In 1991, Dr. Dangelmaier became Professor for Business Computing at the Heinz Nixdorf Institute. In 1996, he founded the Fraunhofer Center for Applied Logistics (Fraunhofer ALB). His e-mail address is <dangelmaier@hni.upb.de> and his Web address is <http://wwwhni.upb.de/cim>.