Coupling of Interactive Manufacturing Operations Simulation and Immersive Virtual Reality

Denis V. Dorozhkin
Virtual Reality Applications Center
Department of Mechanical Engineering
Iowa State University
Ames, Iowa 50011
Email: dorodv@iastate.edu

Judy M. Vance
Virtual Reality Applications Center
Department of Mechanical Engineering
Iowa State University
Ames, Iowa 50011

Gordon D. Rehn
Simulation Group, Industrial Engineering
Deere & Company
One John Deere Place
Moline, IL 61265

Marco Lemessi
Simulation Group, Industrial Engineering
Deere & Company
One John Deere Place
Moline, IL 61265

This is a manuscript from *Virtual Reality* 16 (2010): 15, doi:10.1007/s10055-010-0165-7. Posted with permission. The final publication is available at Springer via http://dx.doi.org/10.1007/s10055-010-0165-7.

Abstract

This paper presents a novel general-purpose simulation analysis application that combines concurrent operations simulation with the advanced data interrogation and user interaction capabilities of immersive virtual reality systems. The application allows for interactive modification of the simulation parameters, while providing the users with the available simulation information by effectively placing the operator in the midst of the environment being simulated. The major contribution of this research is the total integration of the immersive virtual reality environment with the simulation, allowing users in the environment to interactively change the inputs to the simulation as it is running. Implementation and functionality details of the developed application are presented. The experience of using the application to analyze a manufacturing operation in a collaborative scenario is also discussed.

Keywords: Concurrent operations simulation • Virtual reality

1 Introduction

The use of simulation models for the analysis of manufacturing operations is growing worldwide. Simulation has been shown to be an effective tool that can determine the impact of one system component on another, and, as a result, identify manufacturing issues early in the design process in order to avoid unnecessary capital investment and significant rework of a manufacturing process.

Most simulation models are linked to animation tools, which permit the analysis and evaluation of system performance and simulation results in either a two- or a three-dimensional environment. These tools, however, often confine designers to viewing post-processed simulation results using the traditional two-dimensional computer interfaces, such as the monitor, keyboard and mouse, with limited options for making real-time changes to the simulation scenario [1]. Furthermore, although many of the discrete-event simulators do offer the possibility of interaction, they lack the ability to place the user of the simulator in an immersive 3D representation of the simulated scenario [2]. The potential to directly link operations simulation to an immersive virtual reality (VR) environment and to allow users to interactively change the simulation while in process opens exciting avenues for exploring complex interactions between model users, objects and operations being simulated.

1.1 Operations simulation

In its broadest sense, computer simulation is the process of designing a mathematical-logical model of a real system and experimenting with this model on a computer [3]. Discrete event simulation, also known as operations simulation, is characterized by changes in the model state that take place at only a discrete set of simulated time points [4]. Such models rely on the "transaction-flow world view", where the entire system is represented with discrete units of traffic that move between distinct points of the system, while competing for scarce resources. This method is used to determine assembly line bottlenecks, machine tool usage, material handling problems, etc. It is commonly applied to a multitude of industrial and scientific scenarios, including but not limited to manufacturing, transportation, health care, and information processing.

Despite the fact that operations simulation has existed since the early 1960s, industry, in general, has yet to take full advantage of the potential of simulation analysis. In most companies, simulation analysis is used only to plan and verify the most risky or expensive processes. Once a simulation has been created, a team of people are assembled to discuss the facilities, tooling and assembly line issues that result from the simulation. The success of operations simulation relies on the ability of the users to anticipate multiple issues that could occur on the assembly line. Communication between team members with different expertise (tool designers, ergonomists, facilities planning and maintenance, product designers) is crucial in identifying costly errors that potentially would be identified by the simulation. In this research, virtual reality is used to provide a common communication medium to facilitate deep understanding that crosses expertise boundaries between team members.

1.2 Virtual reality in manufacturing operations simulation

Virtual reality is defined as the technology that enables the creation of a computer-generated three-dimensional environment which can be interactively experienced and manipulated by the participants [7]. According to Stuart [8], a virtual environment (VE) is a human-computer interface capable of providing "interactive immersive multisensory 3-D synthetic environments." In these systems position sensors are used to track the user's motions and to update the visual and auditory displays in real-time, allowing the participants to interact with the computer generated environment as if it were the real environment. Interacting in a VE provides all members of the team with the ability to visualize and interact in a natural way by moving around in the environment. The VE removes the traditional interface of keyboard, mouse and monitor, allowing users to easily investigate 3D geometry without having to become experts at manipulating models in the simulation software. With this support, users can more readily draw on their domain-level expertise in product design, tooling design, facilities planning and maintenance, etc. and contribute to the team discussion at a deeper level of understanding.

Several simulation packages that claim to utilize the VR approach to operations simulation currently exist [9, 10]. However, the majority of these programs rely on the traditional two-dimensional (2D) computer interfaces, such as the monitor, keyboard and mouse, to view the 3D models of the simulated environment. Interaction with the environment and the data-interrogation methods remain essentially unchanged from the standard interaction with 2D schematic representation of the simulated manufacturing operations.

One of the few research projects aimed at the investigation of the benefits of a VE in the context of operations simulation was undertaken by Kesavadas and Ernzer [11] at the University of Buffalo. They have developed the VR-Fact program – a virtual environment for modeling and designing factories and shop floors. VR-Fact was created for quick implementation of factory design algorithms ranging from plant layout to factory flow analysis. The program's features were primarily focused on the design of shop floor arrangements using the Cellular Manufacturing (CM) System method. Interaction is supported through the use of a head-mounted display or stereo glasses with a computer monitor. To support team based simulation assessment, this software would need networking capabilities to support multiple users with head-mounted displays or multiple instances of stereo glasses with computer monitors.

Kelsick and Vance [12] at Iowa State University developed a VE which served as a post processor to operations simulation data. The developed program, VRFactory, used results from a commercial discrete event simulation program, SLAM II, to drive a virtual environment animation, implemented in a projection-based VR system with multiple screens. The VE easily supported team discussions of the operations simulation results because of the use of multiple projection screens. Three-dimensional computer models of manufacturing equipment and products were used to allow investigation of how various changes to the manufacturing cell affect part production. Participants would identify different scenarios, run the operations simulation software, and then enter the VE to watch and query the system as time advanced. Changes identified by the team would be fed into the simulation offline, a new scenario would be generated and the team would again enter the VE to examine the new results. The key feature of this application was the ability of the application to read and implement the discrete event actions into the VE, the ability of the participants to navigate to any place within the VE to watch the 3D virtual simulation and the ability of the participants to interactively query any product on the assembly line at any time as to it's status.

Operation of the VRFactory demonstrated that immersion in the virtual factory facilitated the exploration of design changes and their effect on the simulation. The users were also successful communicating among the team members concerning implications of specific design changes on each others expertise domain. The VRFactory program was written as a proof of concept demonstration and therefore it lacked the ability to perform analysis of any other simulation scenario of interest.

The research presented here takes the VE a step further by coupling the simulation and the visualization directly, allowing changes to the operations simulation to occur right in the VE.

2 Simulation framework

This section describes the design framework for the interactive VE to support operations simulation.

2.1 Concurrent simulation

Strassburger et al. [13] identifies four significant features related to the task of coupling simulations and visualizations:

- Temporal parallelism between simulation and visualization;
- Interaction between simulation and visualization;
- Hardware platforms on which the simulation and visualization operate;
- Visualization tool autonomy

These features and the associated options are summarized in Table 1.

Feature	Characteristic	
Temporal Parallelism	**Concurrent** Simulation and visualization run temporally parallel	**Post-run** Visualization run temporally after the simulation
Interaction	**Bidirectional** Simulation and visualization each react to the other tool's commands	**Unidirectional** Only visualization reacts to the simulation's commands
Hardware Platform	**Monolithic/Homogeneous** Simulation and visualization run on one platform	**Distributed** Simulation and visualization operate on different hardware platforms
Visualization Tool Autonomy	**Integrated** Visualization tool is integrated in the simulation tool	**External** Visualization tool works independently of the simulation tool

Table 1 Classification of the simulation/visualization coupling [13].

In this research, to achieve the goal of creating an interactive VE for operations simulation, a temporally parallel concurrent simulation, which is also bidirectionally interactive, is the desired model, as it allows the assembler to observe in real time the effects of his or her actions on the simulation sequence. Furthermore, for flexibility in configuration, a distributed hardware platform will be designed. Finally, since VEs and simulation software have different computational and visualization requirements, the visualization tool will work independently of the simulation tool.

2.2 Simulation Software

The ALiSS (Assembly Line Solution Set) software was chosen as the simulation software; however, the methods described here can be used to interface any simulation software that includes both modules for operations simulation and also for animation. ALiSS is an integrated software package developed by Deere & Company that links a customized user interface with discrete event simulation code and animation software. ALiSS incorporates two commercial simulation software packages from Wolverine Software Corporation™: SLX and Proof Animation. SLX [5] stands for Simulation Language with eXtensibility, and is a classical simulation stand-alone tool that includes a programming language with a C-like syntax. SLX's ultra-fast compiler translates models at nearly 90,000 lines per second, making it one of the faster simulation language on the market. The speed of SLX allows for concurrent simulation/animation execution. Proof Animation™ [6] is a stand-alone animation package. A screen shot from a Proof Animation is shown in Figure 1. Since the display of moving objects in Proof Animation™ is proportional to time, the movement in a Proof Animation™ is smooth and continuous, unlike other animation packages that resort to a 'paint-repaint' method resulting in motion that has incremental jumps in object positioning.

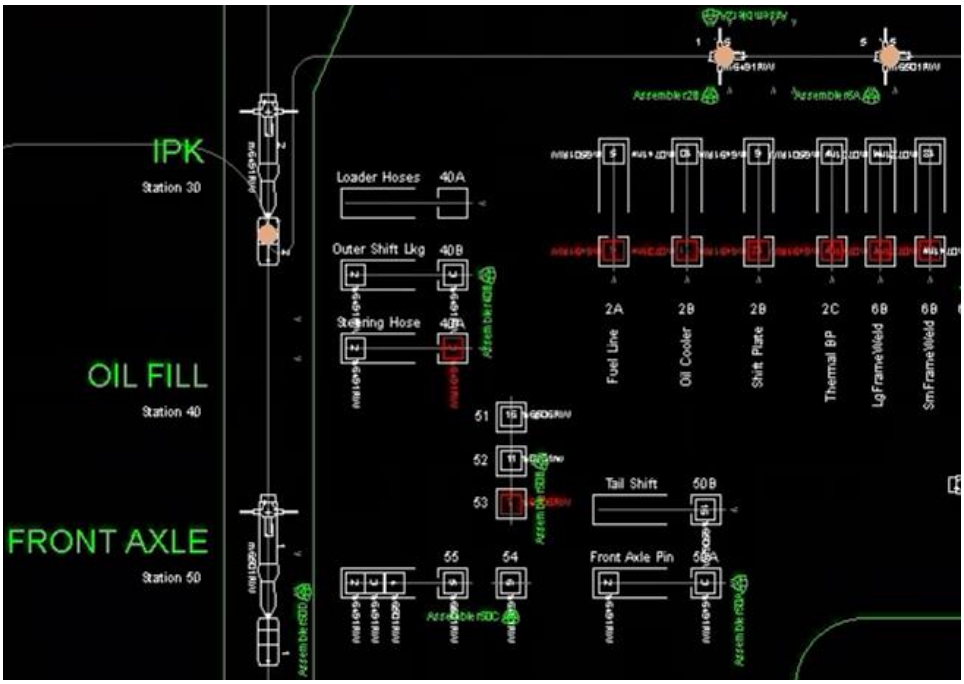

Fig. 1 Sample of a Proof Animation™ scenario.

2.3 Data communication

To address the need for bidirectional communication, a flexible linkage between the outputs of the simulation and the immersive virtual reality environment has been implemented. To accommodate the requirements of a VR-enabled application, a brand new SLX module was developed and integrated into the existing simulation code. The new module is able to extract relevant information from the simulation and pass it to an output concurrent simulation buffer for real-time interaction or a standalone ASCII file (later referenced to as VRF file) for post-processing. The new module does not interfere with the simulation itself, and can easily be deactivated if the specific simulation does not require the collection of relevant information for virtual reality visualization.

Relevant information is extracted by interrogating simulation variables and main active objects (parts assembled, tasks performed, and assemblers working) during any simulation run. Such information includes: time when each assembler arrives and leaves a given working location, or starts and ends a given task; when each part arrives and leaves a given station, or is loaded on or unloaded from a given material handling system; and the status (e.g., busy, idle) of a given assembler, a given part, or a given material handling system, etc.

A procedure has been specifically developed to allow users to choose whether to process all the information relative to a single simulation run, or to limit it to a selected area of the simulated assembly line. Such a procedure reduces memory requirements and improves system performances in the VR system when the VR visualization is not extended to the whole assembly line.

2.4 Simulation interruption

Previous approaches to link virtual reality and discrete event simulation resulted in environments that displayed animations of simulation results. In this research, the intent is to create a virtual environment where the user can modify

the simulation while participating in the animation of the simulation results in the virtual environment. A key aspect to achieving this goal is the development of a method to interrupt the simulation animation, enter additional inputs, and restart the simulation. Synchronizing the simulation and the virtual environment is also a critical task.

According to Strassburger et al. [13], fluent interaction between commercial discrete event simulation software and immersive VE is inherently dependent on the ability to synchronize the two components. Unidirectional time-stepped (equal time steps) or event-stepped (advancement corresponds to actual event-time stamps) logical time advancement are traditionally utilized in the concurrent simulation implementation cases. This, however, can potentially result in the VE exhausting the available simulation data if the discrete event simulation software requires extensive computation time for a given simulation step, forcing a pause in the visualization flow within the environment in order to gain access to new data. One of the ways to address this problem is with standard buffer-based synchronization. In this method, simulation results are accumulated in a buffer waiting to feed into the visualization engine. If the buffer is at its maximum command capacity, the simulation stops the data flow until buffer space is available, thus avoiding any visual delays.

This approach relies on the ability of the simulation to produce data faster than the data can be visualized. However, using a standard buffer approach fails to maintain an uninterrupted animation when the simulation time becomes longer than the time required for visualization/animation since the buffer capacity is based on the *number* of stored commands and does not explicitly control the *time* difference. In the standard approach the time difference varies and is simply equal to the difference between the largest and the smallest timestamp of the buffered simulation data.

To accommodate the unique requirements of concurrent bidirectional coupling of simulation and the visualization software, self-adapting buffers (SAB) were implemented [13]. This is a buffering strategy that adjusts the buffer size based on visualization time intervals. The buffer holds visualization commands within a relatively small time interval, yet contains sufficient number of commands for fluid and continuous visualization. This method supports a variable buffer size that is directly linked to the current visualization speed in the VE, i.e. greater visualization speed results in a larger accumulation buffer time span size.

Using self-adapting buffers, the VE controls the data flow between the system components, not the simulation. The minimum time difference between any two commands stored in the buffer is not enforced, so the total number of such commands can vary greatly. However, the maximum time difference between two subsequent commands, as well as the overall time span contained in the buffer, is strictly enforced. The former is achieved using “dummy” commands that ensure a constant flow of visualization data but do not affect the state of the visualization environment. When a command is deleted from the buffer by the visualization module the remaining time interval is calculated. If it is smaller than the desired buffer size, a supplementary advancement request command is sent to the simulation, identifying the necessary time advance. The ultimate goal is to ensure that the buffer always contains an adequate number of visualization commands necessary for the specified buffer size time interval.

The approach results in a buffer that can properly react to changes in the system, including user interaction, network communication delays and changes in the visualization requirement [13]. Furthermore, it is able to adapt to the current

visualization speed by increasing or decreasing the buffer size time interval with increased or decreased visualization speed respectively.

2.5 Interactive Virtual Reality Environment

The virtual reality simulation program is written in the C++ programming language. Creation of the computer graphics objects is achieved with the SGI OpenGL Performer™, a software development environment that supports implementation of high performance graphics applications and is built atop the industry standard OpenGL® graphics library [14]. The immersive VE is managed with the Open Source VRJuggler virtual reality software library [15]. The application was designed to be used in multi-screen projection-based fully-immersive VR systems. In particular this application takes advantage of the most common hardware configurations of such systems, including wireless wands and tracking systems for interaction with the VE. However, due to VRJuggler's extensibility, the application can be run in any VR system with minimum effort.

The functionality of the VR engine is supplied by the following subsystems:

- *Graphics module*: used for generation and display of the three-dimensional objects in the VE such as the models of the assembly parts and vehicles and the program interface;
- *Data processing module*: responsible for interpretation of the simulation results; it also processes the standard Proof Animation layout files, which contain the motion path information;
- *Logical module*: determines the behavior of the objects in the environment on the frame-by-frame basis, including motion of the parts and vehicles along the paths, animation of the assembly operations, update of the objects' statuses, etc.;
- *Interaction module*: supplies the ability to control the program's functionality with a dedicated VR interface, interrogate the simulation objects, and modify the simulation parameters.

One of the criteria specified for the application was the ability to recreate the assembly environment that is being simulated with the highest level of realism possible. Therefore, actual CAD models of the parts, vehicles and assembly fixtures are used in the VE by the graphics module of the program.

A dedicated model-part association input file is used to identify the particular 3D model that will be used to represent a simulated part or a vehicle in the VE. The file also provides information about the scale of the model and its initial translation and rotation, which could be used to correct for improperly positioned models. This approach ensures that the differences between the physical assembly environment and its virtual representation are minimal. Furthermore, the realistic representation of the assembly workspace makes it well-suited for implementation into the future virtual laboratory for assemblers' training.

The data processing module contains routines for interpreting the VRF and layout files - the primary simulation data files. The layout file contains the description of the paths that will constrain the motion of the parts and the vehicles during the simulation sequence as well as the definitions of the individual segments comprising each path. Motion characteristics associated with individual paths, such as the time it takes for the part or the vehicle to complete the path, or the initial position of the object on the path, are provided in the VRF file.

The VR data stream, produced by the operations simulation code, contains descriptions of all the major events that take place during the simulation time span. Every event or combination of events is preceded by a time stamp, identifying the time elapsed from the beginning of the simulation. During the program's operation each time entry is processed for any events that are to take place. Figure 2 depicts the main execution sequence of the VE program.

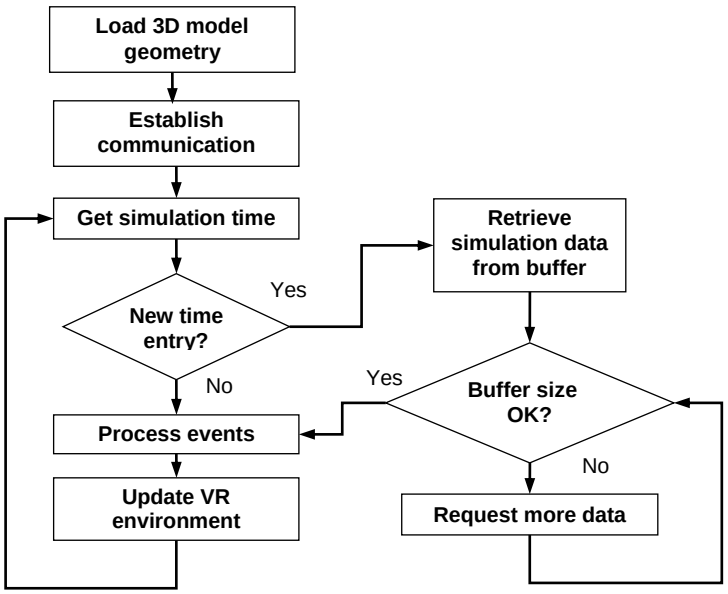

Fig. 2 Application execution loop.

The update of the VE refers to the time spent rendering the scene during each frame of the program's operation. This time is closely related to the graphical state of the environment. The realistic geometry models of the parts, vehicles, assemblers, and the assembly lines increase the immersion factor of the application considerably. It is normally desirable to utilize standard CAD geometry models provided by the industrial partners for visualization of the simulated objects. This streamlines the implementation process and avoids intermediate data conversion steps. However, such models often contain exceedingly high-resolution geometry data, which is not always necessary for realistic representation in the VE. The extraneous data often leads to the reduction in the performance of the application, due to the additional processing requirements.

In order to boost the application's performance to an acceptable level the total number of polygons simultaneously rendered by the system had to be decreased. The intention of this project was to continue using the standard CAD geometry models, provided by the Deere & Company engineering services. Therefore, level-of-detail (LOD) management was developed and implemented. A low resolution model of an object (e.g. vehicle) that has relatively few polygons is normally used. As the distance between the user and the model is reduced, the resolution of the model is gradually increased up to the maximum available level. This approach has resulted in a significant reduction in rendering time and the corresponding improvement of the application's performance.

The amount of time spent rendering each frame is determined precisely using the system clock. That value is then used to compute the current simulation time,

which, in turn, is used as the reference value while processing the current contents of the simulation buffer. Before the next frame is drawn, successive time entries and the associated simulation events in the buffer are processed up to the current simulation time value. The events could include creation and destruction of the parts and vehicles, changes in status of the objects and assemblers, positional placement commands, etc.

The logical module of the application is responsible for keeping the behavior of the simulated assembly objects consistent between the discrete events contained in the data stream originating at the simulation analysis package. For instance, it handles the motion of the parts and vehicles along the paths. The motion of an object in the environment is specified by assigning the object to one of the existing paths, indicating its initial location on the path, and identifying the time it takes for the object to reach the end of the path. From this point in time, the location of the object is updated by the logical module every frame according to the current simulation time, and the object will proceed to the end of the path and remain there until further instructions are provided. This normal procedure is aborted if any additional events associated with the object occur while the object is moving along the path. Such events can include attachment of an assembly part to a vehicle, in which case the motion of the part will be controlled by that of the vehicle, or an encounter of another object on the given path, in which case the motion of all the objects on the path is coordinated in order to keep a certain clearance between the objects.

The logical module also ensures that the status of the assembly objects remains consistent throughout the simulation. For example, the assembly tasks are normally assigned to the assemblers. At that point the program determines which part the assembler is currently working on and updates the part's status accordingly in order to make the assembly task data available in case the part is interrogated by the users of the application. Furthermore, the logical module coordinates the graphical states of the assembly objects according to the current status of the assembly process. For instance, it changes the color of the assembler visualization models to indicate whether the assembler is busy, waiting for the next task, or away from the assembly station. Similar actions are performed in order to correctly represent the parts and the vehicles in the virtual environment.

A set of virtual menus is one of the components of the interaction module. They provide full control over the application's functionality. The location of a menu is determined from the position and orientation of the interaction device, a wireless wand in this case, so that the menu appears attached to the device. This allows the users to keep the menu system from being obstructed by the objects in the environment. The menu system is toggled and navigated with the wand's buttons. Menus can be also used to control the amount of information simultaneously displayed in the virtual environment. For instance, users can toggle on and off all the data labels associated with the assemblers or the parts.

A key component of this work is the ability of the participant to stop the assembly, enter a new time for a task and restart the assembly sequence. Simulation status modification and/or data interrogation of the individual objects in the environment is performed with the wireless wand. By selecting an appropriate menu option and subsequently depressing the wand's trigger button in the immediate vicinity of a part or an assembler, the user can influence the duration of the current assembly simulation task by either terminating the task ahead of the default time or allowing it to proceed past its default termination threshold. These actions are designed to simulate a real-world assembler

completing his or her assigned task(s) ahead of the schedule, or, on the contrary, requiring additional time to carry out the activities (Figure 3). The new time values are then provided to the simulation package and used to reevaluate future simulation events.

The menu system can also be used to toggle the data label containing all the information associated with an object in the simulation environment. Complete range of the simulation data for individual simulation objects (assemblers, parts, vehicles) or for the entire assembly line can be accessed at any time by the users. In case of the part this information includes part type, model type, its unique simulation ID, its status (busy, delayed, or idle), current task performed on the part, and an alphanumeric message that identifies the cause of a delay for the part if such situation occurs. Assembler data tags contain the unique simulation ID, the current task, as well as his/her utilization value – a number (variable between 0.00% and 100.00%) quantifying the ratio between total time the assembler spent working and total time the assembler was allocated to work.

Fig. 3 Interactive assembly task modification.

In order to access areas of the simulated environment located beyond the physical extents of the VR facility, users are provided with the ability to translate in any direction with variable speed. Since some VR facilities lack the rear screen, navigational controls also include the ability to rotate the simulated environment around the current position of the user. This allows the users to investigate areas of the virtual environment otherwise located outside of their visual range.

3 Industrial Case Study

A tractor assembly line was chosen to test the effectiveness of simulation VE. An ALiSS model of the entire line was generated to provide system information and event timing to the virtual reality environment. The assembly line consists of 15 work stations plus over 50 subassembly stations. Served by an overhead

electrified monorail system of 10 carriers, and an automated guided vehicle (AGV) system comprised of 7 vehicles; the line is manned by 28 assemblers working in a single 8-hour shift and flexing between work stations.

The focus of this test is on a single station, which consists of three floor assembly fixtures, tasked with assembling the tractor frame and the tractor transaxle; however, the simulation and visualization method presented here does not impose explicit restrictions on the scope of the simulation scenario. The focus area encompassed the primary components representative of the entire work environment (assembler tasks, assembly part flow, detailed geometry models), and thus was found suitable for the functionality evaluation.

The simulated sequence of events starts by loading an empty monorail carrier with the next transaxle to be built as determined from the current production line-up. Once the main frame assembly is positioned in a fixture, the corresponding transaxle is lowered onto the frame where the two components are ‘mated’ to form the chassis. An overhead bridge crane moves the frame assembly between the three fixtures associated with the workstation. After the assemblers mate the adjoining parts, the entire chassis assembly is raised up to the monorail carrier, that travels to the AGV line where it queues up and awaits unload by the next available AGV. After unload, the empty monorail carrier returns to the transaxle load area.

Fig. 4 Collaborative investigation of the simulated environment.

The simulation framework has been installed at both six-screen (see Figure 4) and four-screen virtual reality systems, and extensive testing and validation has been performed. Several simulation scenarios were investigated, including those with properly allocated and timed assembly tasks and those with artificial bottlenecks in the simulation flow. Special emphasis was made on utilizing the immersive simulation environment for collaborative work with large user groups (5-10 people). Users were able to impact the outcomes of the specific simulation cases by interactively modifying the simulation parameters, creating and/or resolving potential impediments. Based on the feedback from the users, particularly those with no significant background in the operations simulation field, they were able to quickly comprehend the details of the simulated activities and identify the problematic areas on the assembly lines when compared to the traditional visualization methods (i.e., Proof Animation as shown in Figure 1). The experience of using the application to analyze the aforementioned manufacturing operations indicates that it provides good situational awareness and

overall comprehension of the simulation scenario at hand. While a specific assembly line was investigated for the purposes of this research work, the developed simulation framework is capable of analyzing any number of simulation scenarios of arbitrary size, as long as the appropriate data (part and environment models, simulation parameters) is available.

4 Conclusions and Future Work

A method to support interactive manufacturing operations simulation in an immersive virtual environment has been presented. The method was implemented and tested using an industrial application. The overall design advances the state-of-the-art by supporting concurrent (temporally coupled) simulation, rather than using the VE as a post processor to the simulation data. A significant contribution is in providing the users with the ability to easily modify the simulation parameters while immersed in the VE. The simulation can be interrupted, a new time step inserted for a task, and the simulation can be restarted. A method of self-adapting buffers supported synchronization between the simulation and the visualization software guaranteeing a smooth visual animation.

The use of self-adapting buffers serves to cushion the interactive VE from excessive simulation time. The simulation engine steps along and fills the visualization buffer with time-stamped data. Selection of a maximum time for the visualization buffer is currently performed on a trial and error basis. If the VE experiences delays, the visualization buffer time needs to be increased. However, as the visualization buffer time increases, the ability of the participants to effectively halt the simulation, enter a new task time and restart the simulation could get compromised. Additional testing is needed to explore the optimum selection of visualization buffer time for various simulation scenarios to avoid delays in the animation, yet support simulation interruption.

The long-term vision is to develop a Virtual Reality training environment and laboratory for production assemblers. Potential benefits of achieving this vision would include understanding and applying the relationship between product quality, assembler training, and product optionality. To accomplish this goal this project has successfully linked results from an operations simulation application to an immersive Virtual Reality, by establishing a communication link from the simulation directly to the VE. The next step towards a fully immersive assembler training laboratory is effectively inserting support for the assembler to actually assemble the virtual models within the VE. If this could be performed in near real time, the framework would provide an effective testbed for evaluation of work standards in assembly processes.

Acknowledgments

The authors gratefully acknowledge the support provided by Deere & Company.

The authors would like to thank the Institute of Electrical and Electronics Engineers (IEEE) for granting the permission to reuse portions of the article "Integrating Operations Simulation Results with an Immersive Virtual Reality Environment", by Gordon D. Rehn, Marco Lemessi, Judy M. Vance and Denis Dorozhkin, published in the Proceedings of the 2004 Winter Simulation Conference.

References

1. Dessouky, M.M., Verma, S., Bailey, D. E., Rickel, J., *A methodology for developing a web-based factory simulator for manufacturing education.* IIE Transactions, 2001. **33**(3): p. 167-180.
2. Pulugurtha, S., Nambisan, S., Dangeti, M., Kaseko, M. *Simulating and analyzing incidents using CORSIM and VISSIM traffic simulation software*. in *7th International Conference on: Applications of Advanced Technology in Transportation*. 2002.
3. Pritsker, A.B., O'Reilly, J., and LaVal, D., *Simulation With Visual SLAM and AweSim*. 1997, West Lafayette, IN: Systems Publishing Corporation.
4. Schriber, T.J., Brunner, D.T. *Inside Discrete-Event Simulation Software: How It Works and Why It Matters*. in *Winter Simulation Conference*. 1997.
5. Henriksen, J.O. *SLX: The X is for Extensibility*. in *Winter Simulation Conference*. 2000.
6. Henriksen, J.O. *Adding Animation to a Simulation Using Proof™*. in *Winter Simulation Conference*. 2000.
7. Barfield, W., Furness, T.A. III, *Virtual Environments and Advanced Interface Design*. 1995, New York, NY: Oxford University Press.
8. Stuart, R., *The Design of Virtual Environments*. 2001, Ft. Lee, NJ: Barricade Books.
9. Whitman, L., Madhavan, V., Malzahn, D., and Twomey J. *Virtual Reality Model to Aid Case Learning*. in *Industrial Engineering Research Conference*. 2002.
10. Kibria, D., McLean, C. *Virtual Reality Simulation of a Mechanical Assembly Production Line*. in *Winter Simulation Conference*. 2002.
11. Kesavadas, T., Ernzer, M. *Design of Virtual Factory Using Cell Formation Methodologies*. in *ASME Symposium on Virtual Reality Environment for Manufacturing*. 1999. Nashville, TN.
12. Kelsick, J., Vance, J.M., Buhr, L., Moller, C., *Discrete Event Simulation Implemented in a Virtual Environment.* ASME Journal of Mechanical Design, 2003. **125**(3): p. 428-433.
13. Strassburger, S., Shulze, T., Lemessi, M., Rehn, G.D. *Temporally Parallel Coupling of Discrete Simulation Systems with Virtual Reality Systems*. in *Winter Simulation Conference*. 2005. Orlando, FL.
14. Woo, M., Neider, J., Davis, T., Shreiner, D., *OpenGL Programming Guide*. 1999, Reading, MA: Addison-Wesley Co.
15. Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Carolina Cruz-Neira. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *IEEE Virtual Reality 2001 Conference (VR'01)*. 2001. Yokohama, Japan.