Bridging the Gap Between Didactical Requirements and Technological Challenges in Serious Game Design

Stefan Rilling
Institute for Computational Visualistics
University of Koblenz
Germany
rilling@uni-koblenz.de

Ulrich Wechselberger
Institute for Computational Visualistics
University of Koblenz
Germany
wberger@uni-koblenz.de

Prof. Dr. Stefan Mueller
Institute for Computational Visualistics
University of Koblenz
Germany
stefanm@uni-koblenz.de

***Abstract*—Game based training has gained a lively interest within the industry. We present a interdisciplinary work on the application of computer game principles and techniques within an automation industry training scenario. A data flow based component architecture forms the technical foundation of our system and enables us to exploit the capabilities of available middleware components like game- or physics engines. An interactive simulation of a real automation plant is built by the combination of state-based and physical object behavior. A model of training scenarios founded on the didactical principles of game based training is built upon our implemented system.**

***Keywords*-serious games; training simulation; object behavior; data flow architecture;**

I. Introduction

The adoption of virtual technologies has made its way into almost every area of industrial production. The scope of application ranges from planning whole assembly lines to marketing-related objectives. The efficient training and education of employees plays a decisive role within today's globalized industry. Virtual technologies are an up-and-coming instrument to meet these requirements. Using virtual reality (VR) for training has been an important research topic since the first VR systems became available to a broader audience. To this day, a multitude of research projects have dealt with the usage of VR technology for knowledge transfer. One expected a better, more efficient and profound learning performance by exploiting the immersive aspects of VR technology. However, VR systems involve complex and relatively difficultly manageable hard- and software systems like stereo projection, tracking systems and non-standard input devices. This has lead to the fact that to this day, VR systems have not become widely accepted as a tool for education and training.

Besides the field of immersive VR systems, video games have run through a considerable evolution regarding their complexity and visual quality. A multitude of middleware components like game engines or physics engines have been established, shifting the main amount of work developing a modern game from pure programming tasks to content design and production. This has lead to the necessity of taking advantage of video game technology in non-entertaining applications for training and education. These applications are usually referred to as *Serious Games*. The advantage of this technology is its accessibility and its acceptance by a wide user group. Therefore, numerous research projects dealing with Serious Games arose in recent years, and pedagogues as well as computer scientists hope to profit from video games, yet they focus on different aspects of them.

In this paper, we present a system for implementing game-based training scenarios that were designed to comply with various pedagogical requirements derived from an interdisciplinary research project. At first the paper outlines (theoretically derived) technical requirements for serious games and practical lacks of technical frameworks. After that, we describe the organizational context of our work, as well as the learning contents of the curriculum. This section is followed by a detailed specification of our technical framework which aims to meet the foresaid, pedagogically derived requirements. Finally, we exemplary describe how some parts of the pedagogical concept were concretely realized.

II. Related Work

In theory, game-based approaches such as serious games or game-based trainings can benefit from several aspects of video games. However, combining game structure and educational content requires certain design aspects which, in practice, most game engines and technical frameworks lack.

A. Requirements for Serious Game Design

Video games make *systematic use of several didactical princicples* (cf. [1]). Some authors ([2],[3]) argued that because of this, video games are excellent fundaments for training and teaching scenarios. For example, according to Gee, in video games problems are well-ordered, from easy to hard issues, thus providing guidance and supporting problem solving. Also, players train their abilities within repeated cycles of expertise: They repeatedly train a particular skill

("overlearning"), until eventually (usually within a so-called "boss fight") they fail and have to revise their skills, thereby becoming even more competent. In order to benefit from these (and other) principles, serious games have to be designed advisedly, and the technical framework has to be able to realize the game structure exactly as specified by the didactical concept.

Although most video games require a certain amount of instruction, players do not enjoy being taught ([4]). As a result, video games create situations during which players learn something without recognizing. This could, for example, be achieved by non-player-characters ("NPCs"): Depending on the actions of the player, an NPC could subtly focus the player's attention on important aspects of the game situation, provide positive or negative reinforcement or offer a model of successful activity. All these *"stealth teaching"-methods* ([4]), however, require the game engine to comprehend the players behavior – even if the player is helpless and not acting at all.

The *entertainment* deriving from video games is one of the most important, didactical motives for serious games. One particularly important fun factor of video games is game play ([5]). If, in order to motivate learners intrinsically, serious games should take advantage of this enjoyment, they have to combine the game play with the curriculum of the serious game [6]. Therefore, the technical framework has to support game mechanics and game rules in a way that allows the exact realization of the educational game design concept. In many cases, serious games are not stand-alone products, but they are to be *integrated into the on-the-job training* of a company. Usually, such a company already has a lot of learning content resources and an overall training concept. As a result, the technical framework for a game-based training has to be highly flexible in order to match the organizational and technical conditions of the company. Also, the framework should be extendable, thus allowing the companies to reuse and expand their already existing game-based scenarios to fit future needs.

B. Exemplary Problems in Technical Frameworks

According to the previous section, serious games and virtual environments can theoretically have positive effects on training scenarios. Therefore, a multitude of works deals with the application of virtual environments for training and educational purposes. The spectrum of these applications ranges from industrial areas like maintenance and assembly to subject areas from the fields of health care, management, or even military training. However, the full potential of game-based learning can often not be tapped in practice. This section outlines common problems by means of existing systems from various fields of application. For a more extensive overview of existing projects, confer to [7].

The project XVR [1] is a simulation environment intended for the training of rescue- and security forces. Several incidents like traffic accidents, structural fires or riots can be created by a training supervisor. The trainees practice the observation, assessment and decision-making within accident situations while navigating through the simulated world using a joystick. Training sessions are held for a single individual or for whole teams, while an instructor guiding the session is present. The instructor has tools to build incident scenarios and influence running incident scenarios at his disposal.

Another project from the health care and rescue field is *Hazmat: Hotzone*[2][8]. Implemented as a modification ("mod") of the Unreal Tournament engine, Hazmat: Hotzone is a training simulation for rescue forces in the context of biochemical accidents and attacks. The software was created in collaboration with the New York Fire Department (FDNY). Scenario parameters, like the location of the simulated incident, weather conditions, as well as occurring symptoms of the simulated victims are determined by a training manager. A training session combines classical face-to-face teaching by an instructor with a large-scale training mission within the virtual environment. The trainees work in teams, each of the firefighters sits in front of a PC. The trainees navigate through the virtual environment from a first person perspective. They communicate with each other using headsets and other hardware which is used during "real" missions. In Hazmat: Hotzone, all didactical mechanisms are outsourced into the face-to-face part of the training, making the concept more of a blended-learning scenario. Therefore, the virtual environment itself does not need to support didactical principles. Hazmat: Hotzone indeed is an effective training environment, but strictly speaking, it is not a game-based training, for there is not gameplay at all. The overall concept may be effective and the technical framework may be flexible within defined parameters, but it does not meet the requirements of a training concept stressing intrinsic motivation and without face-to-face interaction.

The GVT[3] project [9] originated a development platform for the creation of virtual environments for training maintenance procedures of military equipment. In contrast to XVR and Hazmat: Hotzone, GVT uses a VR System[4] as the primary rendering system. Furthermore, separate libraries for modeling the interaction and behavior of virtual objects (STORM, [10]) and for the description of training scenarios (LORA, [11]) are used. The maintenance procedures consist of a set of sequential subtasks. The GVT platform is not intended to exploit computer game technology. Although complex scenarios can be described with LORA, the requirements of game based training are not completely met with this system.

[1] http://www.e-semble.com/xvr/xvr/

[2] http://www.etc.cmu.edu/projects/hazmat_2005/index.php

[3] https://www.gvt-nexter.fr/Accueil/tabid/150/Default.aspx

[4] http://www.openmask.org/

The Eduventure II[5] project consists of a game which served as a research object for both educational game design and reception. The serious game is a modification of "The Elder Scrolls IV: Oblivion" and has a historical curriculum. The eduventure is an adventure role playing game. The player has to infiltrate a fortress and collect intelligence information during the german revolutions in 1848. The gameplay mainly consists of puzzles and exploration. Curricular contents are closely linked to game structure (i. e. game play, narrative and simulation)[6]. Due to the emphasis of game play and entertainment, the Eduventure II might be considered a more intrinsically motivating educational game. It feels more like a game than an instructional environment. However, the technical framework ("The Elder Scrolls Construction Set") is quite confusing, unstable and has a lot of restraints. Therefore, when it comes to the integration of didactical principles, stealth teaching and flexibility/expandability, the game engine reaches its limits very soon.

III. The AVILUS project

The AVILUS project is a research project funded by the german government department of education and science, leading industry firms and medium-sized businesses. Within this project, we are researching the adoption of principles and techniques from the field of computer games in an industrial environment. The core area of our work comprises training simulations for the automation industry. From a technical point of view, we want to investigate to which extent computer game technology is suitable to implement a realistic simulation of an automation plant and how a virtual training environment can be embedded in such systems. Besides these technical aspects, pedagogical questions mark the other subject area of this interdisciplinary project. The crucial question here is if and how typical elements of game design can be adopted to an industrial training application for engineers, and which consequences for the VR system design would arise from such a game based approach.

A. The industrial environment

We have a real automation testing plant at our disposal which gives us the opportunity to validate the outcome of our developed training environment within a realistic setting. This testing plant simulates procedures of the automation industry and consists of several modules responsible for the filling of small glass-bottles with solid parts, their closure with a cap, and their commission. The movement of the bottles through the plant is realized by conveyor belts, the flow control of the bottles is achieved by the means of track switches, stopping-, and separating stop elements. A single bottle is identified by the plant management system by a bar code which is attached on the outside of the bottle read by a bar code scanner. Bar code scanners are placed at discrete positions within the plant and serve to determine the position of the bottles. The whole plant is not constructed for operation in production, but serves as a platform to investigate and test new technologies within an uncomplicated, flexible and realistic scenery, as well for demonstration purposes and as a discussion basis for the development of innovative concepts. Figure 1(a) shows the equipment.

(a) real plant (b) simulator

Figure 1. Automation plant

The plant is operated by means of a graphical user interface runing on several touch-screen panels distributed over the whole plant. Besides the software user interface, several mechanical switches, buttons and valves exist to control basic functions of the plant like power, pneumatic supply, and emergency stop.

B. The curriculum

The automation plant is a complex device, to operate it, one has to be aware of the position and function of the specific modules, and how these modules are in relationship with each other. In addition, the plant's operator needs to know where the switches and buttons are located within the plant, respectively within the GUI system of the plant's software interface, and in which sequence they have to be activated.

We have chosen the start-up procedure of the plant as an appropriate use case for our training platform to impart the more facile knowledge about the location of components and the sequence of procedures. This procedure consists of three main steps:

- manually switch on the various components of the production line
- activate its control system, connect all stations to the control system via the graphical user interface
- check the station's terminals for messages, activate automatic operation mode.

The start-up procedure suits well to exploit the possibilities of an interactive, three-dimensional simulation of the automation plant. The trainee can virtually walk through the plant, interact with it, and gets a spatial and functional understanding of the machine by receiving visual and auditive feedback to his actions.

[5]http://eduventure.de/viskom_index.php

IV. System description

A. Technical requirements

Several requirements had to be met while implementing the system. One requirement within the project was the ability to exchange specific middleware components without the need to re-implement the whole project. Although modern game engines offer a multitude of functions, it is desirable not to tie oneself down to a certain engine within an industrial environment due to more complex and more frequently changing fields of application compared to video games.

Another requirement resulting from the automation industry scenario is a general description of the behavior of virtual objects which would support the authoring process of modeling an automation plant.

The description of training scenarios is another topic of this work. It should be possible to formulate training goals in terms of user interaction and object behavior. Both the description of dynamic virtual objects and the description of training scenarios should be available in a form on which possible authoring tools can build on.

B. The implemented platform

We formulated a entity-based description methodology in which atomic elements, called *Dynamic Object (DO)* form the basic structural element. In the field of game engines, a entity-based classification of the virtual world is common practice. In our system, a DO is defined as an item in the virtual world, which has a perceivable, distinct behavior, resp. an influence on the virtual world. According to our definition, a virtual world consisting of a room, a table with a fan on it would be made up for example of the DOs room, table and fan. Although the room, consisting of walls, floor, and ceiling is largely static, it is modeled as a DO because it has an influence on the virtual world in a sense that other object are prevented from passing through. On the other hand, if the table were immovable, it could be combined with the room into one dynamic object, as it would not have any distinct behavior. The fan itself can also be decomposed into several DOs, for example into case and fan blades, depending on how independent the several parts are considered.

The DO acts as a container element which encapsulates several *components*. The components themselves comprehend the according functionality. At the current state of development, the object description envisions the partition into *Graphics Components*, *Simulation Components*, *Interaction Components*, *State Components* and *Trigger Components*. The components provide an abstract description of the general data needed to realize each of the components within the run-time environment. The DO as well as each component come with a unique ID used for identification.

Alongside components, *data flows* are the second key elements within our system. Data flows are used to implement communication among the various components of a DO and actually among DOs themselves. For example, the position and orientation of a physics component can be written to the position and orientation of a graphics component using the data flow mechanism. A data flow connects one *input slot* with one *output slot* at a time (c.f. Figure 2), whereas one input- or output slot can take part in several data flows. A component can send data via an input slot, and receive data via an output slot.

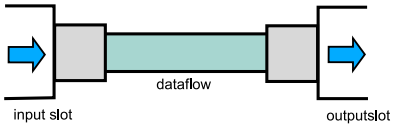

Figure 2. data flow between in- and output slots

Data flows, input- and output slots are identified via a unique ID, so that the input- and output slots of a specific data flow can be denoted by the combination of the DO's ID, the component's ID and the slot ID. This makes the description of the whole data flow network on an ID-basis possible and hence arranges for the extensibility of the architecture.

The execution of a data flow can be subject to several *execution conditions*. Each data flow adheres the activity state of the in- and output slots, and the execution conditions depend on these states of activity. There are four conditions controlling the execution of a data flow:

- The input slot needs to be active
- The output slot needs to be active
- The input slot and the output slot need to be active
- The input slot or the output slot need to be active

Furthermore, the execution of a data flow can occur in a *continuous* or *singular* manner. Continuous data flows are executed with each update step of the runtime environment, singular data flows are executed only once and then deactivated until their reactivation is triggered.

Input and output slots are typed, and only slots of the same type can be directly interconnected. The type determines which kind of data is transported through the data flow. Within virtual environments, the transport of floating point values between different components is one major advent of data flows. Therefore, the system provides slots which transport one to four floating point values. In addition, there is a *void slot*, which does not transport any data, and a *generic slot*, which can transport any type of data. These kind of slots are suited for implementing event mechanisms. Each of the components provide specific input and output slots. A component has to provide an interface which enables the in- and output slots to read, respectively write data with respect to the accordant type of slot.

The graphics component describes the visual properties of the virtual object, such as geometry data, position and ori-

entation. A DO can consist of several graphics components, arranged in a transformation hierarchy. The actual geometry data is not included into the dynamic object description. Instead, the data is referenced by a simple alphanumeric URL, which has to be interpreted by the run-time environment parsing the object description. This mechanism eases the integration of various rendering systems. At a minimum implementation, the graphics component provides input- and output slots for position and orientation.

Simulation Components describe time-discrete simulation objects which receive a position and orientation from their supervising simulation system. Thus, the simulation component also provides input- and output slots for position and orientation. As the physical description plays a key-role in our system, we included a *Physics Simulation Component* specialization of these simulation components. This component encapsulates the whole physical description of a virtual object, including the collision model. The physical component furthermore provides the definition of motion-constraints, e.g. hinge-constraints, which can be found in almost every game-physics simulation package.

Interaction Components describe the behavior of a virtual object in the case of user interactions. User interactions are taking place in the form of *interaction techniques*. An interaction component defines which interaction technique could activate the component and provides an input slot, which gets enabled in the case of a valid user interaction.

A DO's state component represents a set of finite states and the transitions between these states. Our system implements a Petri-Net based state machine, where a place occupied with a token is interpreted as a specific DO's state. Each place provides an output slot which triggers the placement of a token, and an input slot which gets activated by a placement of a token. With this system, state transitions can be triggered via data flows, and data flows can be triggered via state transitions.

The trigger component reacts whenever a DO resides within the trigger's area of influence. The component provides output slots which allow the connection of specific trigger mechanisms within the connected middleware, as well as input slots which get activated as soon as the trigger component notices a DO.

C. Example

Using the system described in the previous section, we are able to model several aspects of the plant's dynamic behavior. In this section, we will exemplarily show the system's possibilities with the plant's filling module, which is responsible for the process of filling the bottles with small solid parts. The device conveys incoming bottles by a wheel shaped device into the filling module by performing a stepwise rotation. The insertion of bottles into the filling wheel is controlled by a separating stop unit, which consists of two stopper elements, alternatingly moving into their base- and working position. The separation unit is controlled by a trigger element. Figure 3 shows a schematic overview.

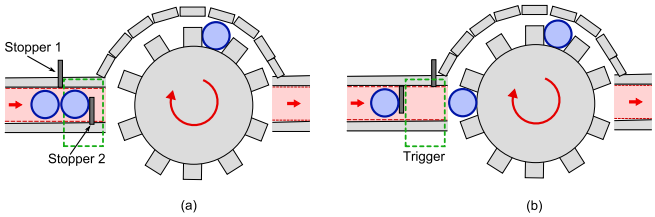

Figure 3. Bottle separation process

The model of the filling module is composed of several DOs, whose different components are interconnected via data flows. The transport system consisting of conveyor belts and the corresponding bearings as well as the simulated bottles are built using standard rigid body and force field components of the physics engine. The filling wheel and the stopper also have a physical representation, so that the whole bottle-transport process is implemented physically. However, a superordinate controll is needed to achieve the desired behavior. We defined separate DOs for each of the two stopper-devices forming the separator unit, the filling wheel and the trigger element. Figure 4 shows an overview of the DO model we used to implement the filling module.

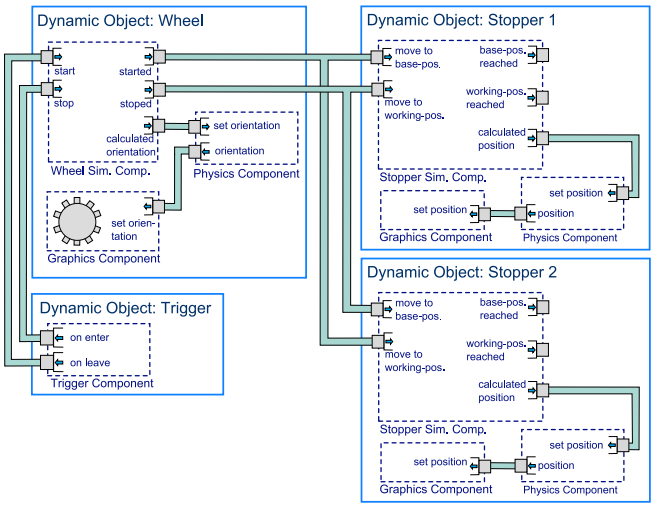

Figure 4. Filling wheel: Components and data flows

For the sake of clarity, we only show the most important components. As it can be seen in the figure, we defined two specialized simulation components for the filling wheel and the stopper. These components are connected to the runtime system via a plug-in mechanism and implement the defined interface of a general simulation component (c.f. section IV-B), showing the possibility to implement own components, resp. the connection of existing components, such as specialized simulation software.

The wheel simulation component controls the step-wise rotation of the filling wheel. Within the components update cycle, the new wheel rotation angle is computed, given a defined rotational speed and a time interval between the rotation steps. The calculated angle is converted to a quaternion and transferred to the physics simulation component via a continuous data flow. The wheel simulation component furthermore provides output slots to start and stop the wheel's motion, and input slots which enable the component to notify when the wheel is started or stopped. The stopper simulation component controlls, similar to the wheel simulation component, the movement of the stopper's physical component to its base- and working position. Being moved into working-position, the stopper blocks the bottle's movement on the conveyor belt. This component also provides output slots to activate the stopper's movement to the accordant position.

As figure 4 shows, the simulation component of the wheel is connected to the simulation components of each of the two stopper DOs, as well as to the trigger DO. The data flows have a singular execution scheme and are executed once their input slots are activated. The defined connection effects that

- the wheel is stopped as soon as a bottle enters the trigger component
- Stopper 1 moves to its base position and hence opens the conveyor belt
- Stopper 2 moves to its working position and hence blocks the conveyor belt

This setup prevents bottles to move into the rotating wheel and corresponds to the behavior of the real plant's filling module.

V. Modeling Training Scenarios

A. Didactical Concept

The game starts out in a bare room. Neither does the player know how he got in there, nor who he is dealing with. A mechanical voice coming from a loudspeaker tells him that, in order to regain his freedom, he needs to pass a difficult test (so far, setting and story background were inspired by the game "Portal" [12] and the movie "The Cube" [13]). In the beginning, the voice directs the player, telling him what to do next. After the player has accustomed himself to new surroundings, the voice withdraws, leaving the player on his own. He has to figure out what he is expected to do and how he has to do it all by himself. However, there are subtle clues and suggestive arrangements. For example, there is an inconspicuous whiteboard showing an abstract plan of the production line, location and visual nature of switches and values etc. (cf. figure 5).

The purpose of these clues is to lead the player without depriving him of gratification. Also, the environment delivers direct feedback on his actions, thereby notifying him of

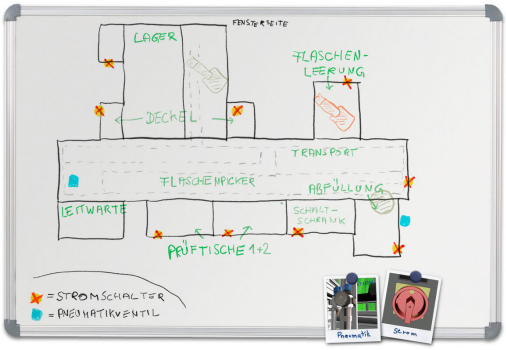

Figure 5. If interpreted correctly, the inconspicuous whiteboard supplies valuable information for the tasks and puzzles

errors and success. Once the player has successfully completed a set of logic puzzles (e.g. finding a certain pneumatic valve), he must prove himself in a "boss sequence" during which he has to cope with a mixture of previous tasks under more challenging conditions (e.g. finding and activating all power switches within a certain time frame). Once he has mastered a boss level, he gets a reward and feels he has come closer to his goal. The player can not die during the game: If he fails a boss level (which, due to the level of difficulty, is likely to happen for the first one or two times), he can just repeat it. Eventually the whole production line would be activated and the player would finally pass the test. He may then leave the game or he may freely interact with the now fully operational production line at his fancy.

If we did things right, the learners should both have fun and efficiently learn to start the production line by playing the game.

B. Technical Realization of the Concept

We implemented a system to describe and control training scenarios within the virtual environment. The *training control system* builds on the dynamic object description. During a training session, the trainee has to complete several tasks. Completing a task implies interaction with the environment, and interaction with the environment implies either navigation through the environment or manipulation of the environment. Within the context of our DO description, manipulating the environment (and hence the DOs) changes the properties of the DO's components. Therefore, one can express the meaning of a task by internal parameters of the components of one or several DOs. This means that the author of a training scenario specifies for each of the scenario's tasks the involved DOs and the target-values of the specific DO's components. The concrete occurrence of these target values depends on the chosen type of component. For a DO's state component, this value would be a specific state the component has to reach. For a physics component, this value could be a certain position or orientation the DO has

to reach so as to complete the task.

We model this atomic object interaction as an *action description*. The action description controls exactly one specific component of one specific DO, i.e. an action description of a state change would observe the state component of the assigned DO. A task can control several action descriptions and is considered as fulfilled as soon as all the required interactions are performed.

We distinguish tasks by the processing of their action descriptions. There are tasks where the processing order of the action descriptions is either relevant or irrelevant, we call these tasks *random tasks* and *sequential tasks*. Sequential tasks can be further distinguished by the following criterion: Either all options for action are available right from the beginning or unlocked during the task. In the first case, action descriptions executed out of order can be considered as errors and the training control system would execute the defined reaction, in the latter case, the training control system would attend to the activation of the specific components in the correct order. By this categorization of tasks, we can model tasks following the principles of overlearning and well-ordered difficulty (cf. sec. II-A).

In order to keep a training session motivating and suspenseful, and to support the active learning principle of game based training, the training system has to provide functionality to meet these requirements. The possibility of making errors as well as having to deal with time pressure are two key-elements of this didactical principle. Therefore, the training system provides the possibility to explicitly specify an erratic behavior of the trainee. This is achieved by marking action descriptions of a task as failing. In the case of an erratic behavior, the training control system can be configured to either reset the task, forcing the trainee to repeat the lesson, or to just monitor the errors for a later analysis.

Several tasks of a training scenario can be summarized to a *task group*. The tasks inside a task group usually belong to one dedicated educational objective and can be compared to the concept of "levels" one can find in computer games. Figure 6 shows the structure of our training scenario

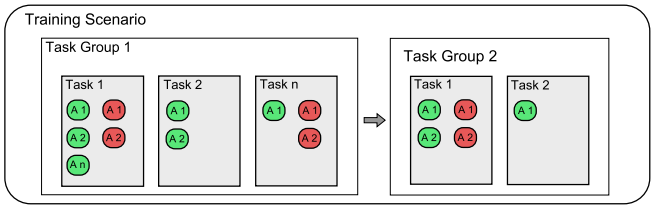

Figure 6. structure of training scenario

model with its task groups, tasks, and action descriptions. The failing action descriptions are colored red, the action descriptions which are needed to accomplish the task are colored green.

The connection of the training control system to the virtual world is realized using the component and data flow mechanism of the dynamic object runtime environment. Our system currently envisions action descriptions which monitor state- and physics components of a DO. As the querying of a specific state of an object might be an obvious choice for monitoring an object's behavior, querying the physical properties of a DO can be a useful tool to express spatial tasks, where, for instance, an object has to be put to a specific position.

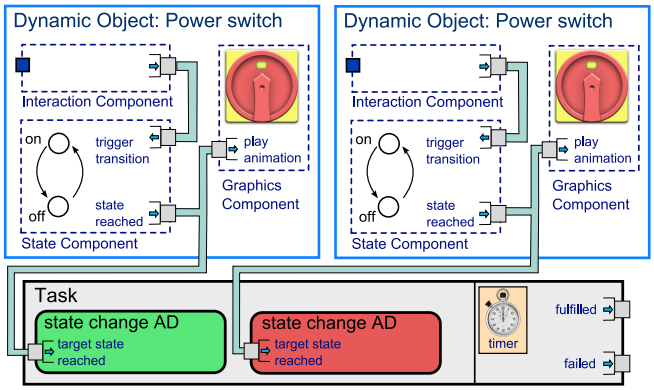

Figure 7. connecting tasks to the environment

Figure 7 shows the combination of action descriptions and the dynamic object description using the data flow mechanism. Within the figure, a training task comprises the operation of one power switch, where it is possible to activate the wrong switch. The state change action description provides output slots which can be used to establish a conditional data flow between the corresponding input slot of the state component and the action description.

C. Example

We have modeled the startup procedure of the plant using the described model. In this section, we will show exemplarily one part of the whole procedure, namely the opening of the two air pressure valves and the engaging of the various stations power supply. This scenario is described by a task group containing two random tasks, the first task represents the opening of the pressure valves, the second task represents the operation of the power switches. The two tasks show an increasing level of difficulty and pressure. The first task, opening the pressure valves, is intended to acquaint the trainee with the controls and the navigation within the virtual environment. The trainee cannot fail and has an unlimited amount of time available, all other interactive objects are disabled. Once the first task is finished, the training control system enables the second task, which is slightly harder to fulfill, as this task is bound to a temporal constraint. The trainee could fail if all the power switches are not turned on within the given time frame, in this case the training control

system resets the virtual world to its initial state and the trainee has to repeat the task.

The whole scenario meets the requirements for serious game design (c.f. II-A), particularly the aspects of overlearning and increasing levels of difficulty.

VI. Conclusion

In our approach, we try to tap the full potential of computer games within an industrial training context. Therefore, we consider serious games from both a technical and a didactical point of view.

On the *technical* side, our system supports the connection to established middleware components like game- or physics engines. The defined system architecture allows the replacement of components, respectively the embedding of own components. We combine state-based object behavior with a rigid body simulation system to a lifelike and convincing virtual model of an existing automation plant.

On the *didactical* side, all pedagogical requirements, defined by a proof of concept-scenario, were met. We implemented a training scenario for the whole start-up procedure of the plant and were able to realize didactical principles, stealth teaching and authentic game play. Also, the system fits in the on-the-job training concepts of our industrial partner.

Within this interdisciplinary work, we have shown the possibilities of the application of computer games in an industrial training scenario. Compared to classical VR systems, computer game technology eases the creation of interactive virtual worlds with rich object behavior. However, in spite of all technological advantages, technology alone is just one part in the field of serious gaming. The pedagocial principles of game based training raise their own particular requirements to the underlying technical systems. In conclusion, our technical framework is geared to didactical needs regarding serious game design. Now that we know we can implement the didactically important aspects of a serious game concept, our future work will concentrate on developing, realizing and evaluating two game-based trainings (making systematic use of the didactical aspects mentioned in section II-A) for our partner. These will be completed and evaluated by late summer of 2010.

References

[1] P. J. Gee, *Good video games + good learning: Collected essays on video games, learning, and literacy*, ser. New literacies and digital epistemologies. New York: Lang, 2007, vol. v. 27. [Online]. Available: http://www.loc.gov/catdir/toc/ecip077/2006101455.html

[2] R. van Eck, "Digital game-based learning. it's not just the digital natives who are restless," *Educause review*, vol. March/April 2006, pp. 16–30, 2006. [Online]. Available: http://www.educause.edu/ir/library/pdf/erm0620.pdf

[3] D. A. Gentile and J. R. Gentile, "Violent video games as exemplary teachers: A conceptual analysis." *Journal of Youth and Adolescence*, vol. 37, no. 2, pp. 127–141, 2008. [Online]. Available: http://www.springerlink.com/content/7706114365625653/

[4] M. Bopp, "Didactic analysis of digital games and game-based learning," in *Affective and Emotional Aspects of Human-Computer Interaction*, M. Pivec, Ed. Amsterdam: IOS Press, 2006, pp. 8–37.

[5] H. Wang, C. Shen, and U. Ritterfeld, "Enjoyment of digital games. what makes them "seriously" fun?" in *Serious Games. Mechanismms And Effects*, U. Ritterfeld, M. Cody, and P. Vorderer, Eds. New York: Routledge, 2009, pp. 25–61.

[6] U. Wechselberger, "Teaching me softly: Experiences and reflections on informal educational game design," *T. Edutainment*, vol. 2, pp. 90–104, 2009.

[7] S. de Freitas, "Learning in immersive worlds. a review of game-based learning," Joint Information Systems Committee, 2006, available online at http://www.jisc.ac.uk/eli_outcomes.html; visited on April 2010.

[8] S. Carless, "Postcard from sgs 2005: Hazmat: Hotzone - first-person first responder gaming," Gamasutra. The Art & Business of Making Games, 2005, available online at http://www.gamasutra.com/features/20051102/carless_01b.shtml; visited on April 2010.

[9] S. Gerbaud, N. Mollet, F. Ganier, B. Arnaldi, and J. Tisseau, "Gvt: a platform to create virtual environments for procedural training," in *VR*, 2008, pp. 225–232.

[10] N. Mollet, S. Gerbaud, and B. Arnaldi, "Storm: a generic interaction and behavioral model for 3d objects and humanoids in a virtual environment," in *13th Eurographics Symposium on Virtual Environments*, 2007, pp. 95–100.

[11] N. Mollet and B. Arnaldi, "Storytelling in virtual reality for training," in *Edutainment*, 2006, pp. 334–347.

[12] V. Software, "Portal," 2007.

[13] "Cube," 1997.