GVT: a platform to create virtual environments for procedural training

Stéphanie Gerbaud, Nicolas Mollet, Franck Ganier, Bruno Arnaldi, Jacques Tisseau

GVT: a platform to create virtual environments for procedural training

Stéphanie Gerbaud* IRISA-INRIA-INSA de Rennes
Nicolas Mollet† INRIA-IIT
Franck Ganier‡ CERV-UBO
Bruno Arnaldi§ IRISA-INSA de Rennes
Jacques Tisseau¶ CERV-ENIB

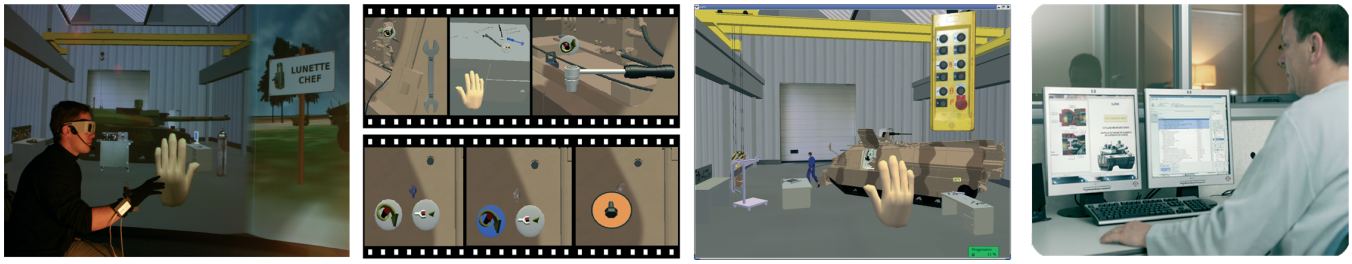

Figure 1: A trainee, who interacts with objects to perform the procedure, within the virtual environment, supervised by the trainer

ABSTRACT

The use of Virtual Environments for Training is strongly stimulated by important needs for training on sensitive equipments. Yet, developing such an application is often done without reusing existing components, which requires a huge amount of time. We present in this paper a full authoring platform to facilitate the development of both new virtual environments and pedagogical information for procedural training. This platform, named GVT (Generic Virtual Training) relies on innovative models and provides authoring tools which allow capitalizing on the developments realized. We present a generic model named STORM, used to describe reusable behaviors for 3D objects and reusable interactions between those objects. We also present a scenario language named LORA which allows non computer scientists to author various and complex sequences of tasks in a virtual scene. Based on those models, as an industrial validation with Nexter-Group, more than fifty operational scenarios of maintenance training on military equipments have been realized so far. We have also set up an assessment campaign, and we will expose in this paper the first results which show that GVT enables trainees to learn procedures efficiently. The platform keeps on evolving and training on collaborative procedures will soon be available.

Keywords: virtual environments, virtual reality, training, procedure, behavioral objects, collaboration

Index Terms: D.2.13 [Software Engineering]: Reusable Software—Reuse models; I.6.5 [Simulation and Modeling]: Model Development—Modeling methodologies; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; K.3.1 [Computers and Education]: Computer Uses in Education—Collaborative learning;

*e-mail: stephanie.gerbaud@irisa.fr
†e-mail: nicolas.mollet@iit.it
‡e-mail: franck.ganier@univ-brest.fr
§e-mail: arnaldi@irisa.fr
¶e-mail: tisseau@cerv.fr

IEEE Virtual Reality 2008
8-12 March, Reno, Nevada, USA
978-1-4244-1972-2/08/$25.00 ©2008 IEEE

1 INTRODUCTION

Training is a crucial need for industries. Virtual reality offers many assets for such training: lower costs and risks (for humans as well as for equipments), no need for (often costly) physical equipments, possibility to train on yet undeveloped equipments, new pedagogical actions which are not possible in the real world, easy monitoring of trainees' activity. Yet, the efficient development of complex virtual environments is a major issue. We aim at developing a full authoring platform for building virtual training applications. For this purpose, we propose generic and reusable approaches which can be used in virtual environments in a general way.

Training in virtual environments can have various objectives. Namely, in our applicative context: maintenance training, the main need is to learn maintenance procedures rather than technical gestures, already mastered by technicians. The reference procedure is therefore the central part of the Virtual Environment for Training (VET). It expresses what can be done at each step, and must be strictly respected and fully completed by the trainee.

We have developed GVT (Generic Virtual Training)[1]: a platform to create and execute virtual environments for procedural training. GVT is intended to offer monitoring facilities to the trainer and to facilitate the creation of a new training session. This last point is ensured by the capitalization of the created data (scenarios, objects behaviors, etc) and the use of authoring tools.

After analyzing the requirements for a VET and presenting a state-of-the-art, we will describe the GVT platform: the main models, the global architecture and the authoring tools. Then we will present the GVT project and its actual release, followed by the work done in order to add collaboration capabilities to our platform. Finally we will expose experimental results about procedure learning within the GVT environment, before concluding.

2 RELATED WORK

2.1 Existing Virtual Environments for Training

The models used in existing Virtual Environments for Training are rarely generic enough to suit various VET requirements from various applicative domains. Nevertheless, all VETs have common needs and the well-known STEVE project [14] is a good example to illustrate those needs. The initial aim was to propose individual virtual training on the maintenance of a boat. The virtual environment was managed by a simulator (VIVIDS), which was used to simulate objects behaviors. The authors developed a pedagogical

[1] http://www.gvt-nexter.fr/

agent, named STEVE, which supervises the objects, the actions, and the procedure to perform. This procedure was modeled thanks to a task approach, with constraints on actions. The STEVE project has then been extended to support team training. With this illustrative example, we can identify general needs of a virtual training environment: a reactive environment composed of behavioral objects, a description of the procedure to achieve, and pedagogical tools for the supervision of trainees. We do not aim at describing in detail the pedagogical aspects of our platform, they will be shortly described in the description of our platform architecture. In this paper, we rather focus on the first two needs of a VET: the virtual environment and the procedure which are detailed next. We will then focus on a more precise category of VETs: those which allow collaborative training, and we will expose their specific requirements.

2.2 Behavioral objects and interactions

The first need we have identified for a VET is to offer a reactive environment, composed of behavioral objects. As trainees have to interact with objects, we have to describe how to realize the 3D animation, how to model the relationship which exists when a trainee takes an object in his virtual hand (the relationship between the user model and the object), how to model the reaction of a virtual electronic tool when a user presses some of its buttons, etc. Those questions orient our state-of-the-art on how to define the behavior of potentially complex objects that may be encountered in an industrial maintenance procedure, and how to describe potentially complex interactions between objects.

Research focused on modeling the behavior of 3D objects is part of the field of behavioral animation. In this area, we can find three main approaches: stimulus response systems [36], rule systems (Ex: Reynold's flock of bird [27], Tu's virtual fishes [35]) and state machine systems [6][4]. Stimulus response systems are interesting to obtain reactive, adaptable and reusable limited behaviors, but such approaches have a low level of abstraction and do not propose a fine control of the obtained behavior. Rule systems offer a higher level of abstraction, but when the number of rules greatly increases, deductions can take much time, which is a major drawback in a real-time application. Furthermore, the reusability of such a system is limited as it is necessary to reuse the whole rule system. State machine systems can appear less easy to use because they do not allow a declarative description of the behavior whereas rule systems do; but the use of advanced state machines allows a fine control without long deductions, and increases the potential reuse of several behaviors. Therefore, modeling the behaviors of 3D objects with advanced state-machines appears to be an efficient solution which allows complex descriptions and the re-using of behaviors and objects.

Complex interactions need a high level of abstraction to be easily defined, and those problems have been mainly formalized in works on virtual humans. Interactions of a virtual human with behavioral 3D objects can be recorded and replayed: it is a classical method which has been largely used in video games. Yet, the reusability of such an approach is limited as it requires a lot of data acquisitions. Moreover the replay has to be prepared by pre-positioning the virtual character and objects in interaction. Another approach is based on Artificial Intelligence techniques. This top-down view is based on human beings. For example, to interact with a door, a human being looks for the door handle and then uses his memory and deduction to choose what to do and how. Hildebrand [12] proposed an application where a user can ask a virtual character to interact with objects in a virtual environment. The states and behaviors of the virtual character and the objects are managed by an expert system. Although this system is really interesting, due to real-time constraints, it does not presently offer complex and varied interactions with many objects and many configurations. The last family of interactions is based on informed environments. In those approaches, behavioral 3D objects of the virtual environment contain a description of what they can propose as interaction or how to use them. Kallmann [15] has proposed a major contribution in this field: smart objects. Informed environments appear to offer more flexibility and reusability than other approaches. Nevertheless, there currently is a problem of particularization of the virtual human, which does not allow describing complex interactions between complex behavioral objects in a general way.

2.3 Scenario description

The second need we have identified in our industrial context on maintenance procedure training is a description of the procedure to achieve: how can we describe complex and long sequences of actions in a virtual environment, composed of complex behavioral objects and virtual humans? In order to propose a solution, we have to analyze how we can describe scenarios in virtual environments in a general way.

We can define two main families of approaches to describe procedures in this context. The first approach for arranging actions in a virtual environment is based on a low-level approach. As some tools allow the definition of complex behaviors, they can be used to describe complex scenarios (for example HCSM model in [4]). There are two main drawbacks: the low level of abstraction and the use of computer languages which are not accessible to non computer scientists. Therefore, another level of abstraction is needed, that is the role of the second main family of approaches: scenario languages. We can define three main approaches in this family. First, the use of interactive stories [23]. This method is interesting to describe a complex variable story with multiple characters, and to train on decision-taking. But interactive stories are less interesting when the goal is to describe a fixed story, like a maintenance procedure. Second, we can find the full description of the sequence of actions (Ex: Q language [13], Grafcet [19]). We can extract interesting properties from those scenario languages, such as hierarchical concept, parallel actions, parameterization, adaptation to non computer scientists (for example Grafcet is a graphical language). Third, with a more declarative approach, scenarios can also be managed with constraints and goals [25][14][3].

Describing scenarios with constraints and goals is more flexible than a complete description of sequences. It allows for example the readaptation of the scenario when a non-predicted event happens. Nevertheless, this flexibility is a problem when the aim is to learn an exact procedure: the number of constraints and goals have to largely increase in order to be sure to produce the only solution required by the industrialist. As a consequence, a full description of the sequence of actions is more adapted to industrial maintenance procedures, where the industrial solution is the only solution to follow. Yet, existing works do not allow to easily express complex and common sequences of actions that may appear in a procedure such as a maintenance procedure. We must also take into account the fact that describing scenarios is generally a hard task for potential authors of scenarios, as they are experts in training procedures and not computer-scientists. Therefore, we should offer them a graphical language.

2.4 Collaborative virtual environments for training

Among VETs, some of them offer collaboration capabilities. We can distinguish between three types of collaboration: collaboration between real users (RU/RU), collaboration between one real user and one virtual human (RU/VH) and collaboration between virtual humans (VH/VH). Therefore we can classify existing collaborative VETs into three categories.

- VETs which offer RU/RU collaboration only. In these environments, we only have real users who train together (Ex: COVET [22] or the VET presented in [7])

- VETs which offer RU/VH and VH/VH collaboration. In these environments, we have only one real user (the trainee) who collaborates with several virtual humans. Most existing collaborative VETs enter this category (Ex: MRE project [34, 32] or SECUREVI [24])

- VETs which offer the three types of collaboration (RU/RU, RU/VH and VH/VH). In these environments, we can have several real users and several virtual humans. (Ex: the extension of STEVE [14] to support team training [28, 29])

This third category is the one we are interested in since this type of collaboration is the most difficult one. Indeed, in such a VET we must deal with both virtual humans with a predefined behavior and real users whose actions are unpredictable.

Moreover, we have identified three major additional functionalities for a collaborative VET compared to a non-collaborative one. First, a user must be aware of the others and of their actions. Secondly, we must be able to describe the collaborative procedure to realize and to identify who can perform each action. Lastly, virtual humans should be proposed to replace missing team members or to make the trainee collaborate with team mates with a specific behavior. We have also detected some limitations among existing collaborative VETs. First, to identify who can perform each action, users are assigned a role and this role is matched with the one allowed for each scenario action. Thus, only one role is allowed to realize one scenario action, which results in a fixed distribution of actions between the team members. Moreover, virtual humans often follow the procedure without making any mistakes or without acting in an unexpected way.

3 THE GVT PLATFORM

Although various models exist to create behavioral objects and to describe scenarios, none of them fully satisfy our needs. As a consequence, we propose a set of models and solutions in order to achieve the efficient development of VETs and the reusability of their components. Those models have been developed and validated in the GVT industrial project detailed in the next section.

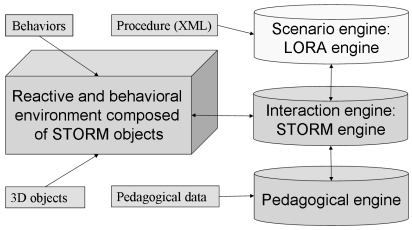

Figure 2: Global vision of our platform

The analysis of the needs of a VET and the state-of-the-art lead us to define the following global architecture for our platform (on Figure 2 we can see the main components and the workflow between them). This global architecture relies on two main models that we will present now: a model for behavioral objects and interactions, STORM, and a model for procedures, LORA. Then we will describe how they are integrated into the global architecture of the platform. Finally, we will explain how to create a new training session, thanks to authoring tools that rely on these models.

3.1 Models

The GVT platform relies on two main models already published: STORM [21] and LORA [20]. Therefore, we will only give a short outline of their key concepts and their interesting properties in the following sections.

3.1.1 STORM

We mixed different interesting properties found in our sate-of-the-art in order to propose efficient and innovative solutions to model behavioral objects as well as the interactions possible between them. We designed the STORM model with a view to improve the reusability of objects and interactions and to generalize the concept of objects to the virtual human. STORM (Simulation and Training Object-Relation Model) is defined in two parts, a model of behavioral object and a model of interaction.

A STORM object is composed of internal private activities and is endowed with a set of capabilities. A capability represents a possible interaction that the object can offer. It is made up of two elements: a public activity (the behavior and the object reaction) and an associated interface (a standard protocol of communication which allows the object to communicate with other unknown STORM objects). Even if most of the activities of our objects are modeled using HPTS++[17] (Hierarchical Parallel Transitions System ++), a language of hierarchical automatons, STORM does not impose the mean of implementation of a behavior. To create a new STORM object, we simply associate different capabilities of interaction. For example, a plug has got a male-screwing capability which offers particular characteristics (such as the size, states, etc.) to the object.

The STORM relation represents the link that exists when objects are interacting. This specific STORM object contains all the information required to define an interaction. A relation uses the capabilities of the objects to create the interaction between them. For example, a screwing relation will use both male screwing capability of a plug and female screwing capability of a female support, to make the screwing link between those two objects.

We can list some interesting properties of this model:

- Objects behaviors can be very complex, as the model doesn't impose how the behavior has to be realized. A STORM object simply has to offer capabilities, which allows other objects to interact with it. It is possible to adapt previous behavioral objects developed without STORM: those objects just need to be endowed with the corresponding capabilities.

- The local processing of interactions is stable and dynamic. It also is easy to modify and maintain.

- Objects can easily be increased with reusable interaction capabilities. STORM objects can be directly reused.

- There is no particular object in the virtual environment. In comparison with the state-of-the-art, here avatars and virtual humans are objects interaction capabilities, like any other behavioral object.

- Finally, it is important to underline that a relation is a STORM object, so that it can supervise other relations: the interaction process is generic at each level.

As an example of use, we simulate the evolution of pressure between two complex hydraulic objects that trainees have to screw and to connect with dedicated tools. The different 3D animations are triggered by the STORM relation which also manages the physical link with the management of the pressure conducted in a hydraulic network composed in real-time by trainees. More information on the STORM model can be found in [21].

3.1.2 LORA

In our industrial context, procedures and in particular maintenance procedures are very strict (actions have to be performed in exactly the given order), long and complex. Indeed, our industrial partner has a full cupboard of maintenance procedures descriptions for their equipments. Our goal was to allow non computer scientists to

transcribe these procedures from their textual version to a computerized one. Therefore we allow authors to completely describe the procedure, without necessarily making the causality explicit thanks to preconditions and goals. Indeed, in such procedures, causality may never be described, and the exact industrial procedure is not always the only solution for the local set of constraints and goals. Yet our scenario model does not prevent from describing causality; preconditions can be added in order to improve the flexibility of the scenario when needed. As an example, in section 5 we introduce preconditions on the tools required to perform an action.

We proposed a scenario language, both graphical and textual, named LORA (Language for Object Relation Application). Its philosophy is to describe what can be done at each point of the scenario. It is inspired by different graphical languages such as Grafcet, and different hierarchical and parallel state-machines: LORA is a hierarchical parallel transition system language. It gathers a set of hierarchical state machines, composed of a set of variables, steps and links between steps. A state machine has a current state: a set of active steps which represent what could be done as long as they are active. The scenario engine is a multi-agent system which dynamically interprets the scenarios during the Run-Time execution. Therefore, the scenario can be modified in Real-Time. This engine also offers a service of resources management: it tests whether the objects are available and whether current actions can be achieved with them, and manages state machines for concurrent access to the same objects.

We can list some interesting properties of this scenario language:

- The graphical approach allows non computer scientists to author complex scenarios, with for example parallelism or exclusivity between tasks. It also allows the easy graphical-writing of common conditionals.

- The description of what can be done by a trainee is more flexible than a what-must-be-done approach.

- The engine ensures actions can be achieved: it offers a resource management on objects and state machines. This also contributes to the simplicity of scenario authoring.

- The hierarchical approach allows for flexibility in the description and reusability of part of scenarios, while state machines can be parameterized with variables on objects and actions.

- As the engine operates like a virtual machine which interprets the scenarios, we can create, modify and adapt procedures in real-time, during the simulation. This is a fundamental characteristic for interactive authoring tools.

More information on the LORA language can be found in [20].

3.2 Architecture: the kernel of our platform

The analysis of the state-of-the-art led us to propose a kernel based on four elements (Figure 2):

- A reactive and behavioral environment. By using the STORM model, we create a reactive and informed 3D world, made of various behavioral objects which can interact with one another.

- An interaction engine. This engine describes what could be done in the environment: what are the possibilities of interaction with available objects. The interaction engine is the STORM engine. The role of this element is to manage interactions between STORM objects, depending on their states and on the interactions they offer.

- A scenario engine. This engine describes what could be done by trainees: what are the possibilities of interaction at the current state of the procedure represented by LORA. As actions are performed in the environment, the engine evolves and updates the possible next steps.

- A pedagogy engine, the trainer assistant. This engine uses the two engines above, to choose what trainees will be allowed to do. This engine has to react to trainees actions and tailors its reaction to each trainee's particularities, such as his familiarity with the procedure (ex: a novice needs to be guide). Many pedagogical strategies can be developed here.

Those elements compose the heart of our platform of development. All objects and behaviors designed with the STORM model are totally reusable. The STORM engine does not need any special configuration, it only uses the informed environment. The pedagogy engine can use the scenario engine and the trainee's characteristics to automatically create its strategy of response and action. Concerning the scenario engine, parts of scenarios can be reused in other training sessions when they represent common sequences of actions.

3.3 How to create a training session: authoring tools

All the data of the environment (objects, capabilities, procedures, etc.) can be created and modified manually. But in order to efficiently create, modify and maintain new training sessions (including for non computer scientists), we worked on the idea of intuitive tools for interactive prototyping, by using the dynamic properties of our models and engines.

A set of internal authoring tools is now at the author's disposal in order to create the 3D environment (to position the objects), to design STORM objects (or reuse existing ones) and endow them with capabilities and to define pedagogical strategies. Besides, these tools provide an access to data libraries (3D objects, capabilities, pedagogical actions) in order to reuse previous developments and facilitate the author's work. Once the virtual environment is created, the author can specify the scenario. An innovative tool, named "building by doing", enables to describe the procedure, directly within the GVT environment, at the runtime. Instead of writing the procedure in XML, respecting the syntax of the LORA language, the author simply realizes the procedure. This tool displays two windows (Figure 3): the first one is a view of the GVT environment, exactly as the trainees could have; the second window is a 2D interface. As the author demonstrates the procedure in the first window, the tool records the actions made, generates the corresponding steps in the LORA language and displays them in the second window. In the 2D interface, the tool supplies a direct access to the generated procedure, thanks to the interpreted aspect of LORA, in order to specify particularities: to make new branches, to change the arrangement of actions in the procedure, to modify the parameters of an action, to create conditions or macro-steps, etc.

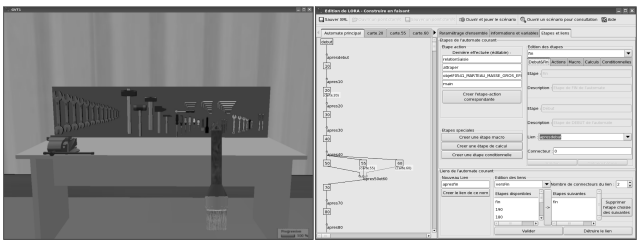

Figure 3: The "building by doing" authoring tool

The "building by doing" tool is integrated as a plug-in of the kernel. The tool can record every action realized in the environment

with a direct connection to the STORM engine. A 2D interface offers the access to the memory of the scenario engine. It also offers a graphical visualization of the current scenario in the graphical version of LORA. Therefore, the new steps generated by the actions can be automatically included or modified.

Thanks to this innovative tool the author does not have to program the scenario anymore, he just needs to show it. This authoring tool supplements the other ones and enables us to provide a complete set of tools to easily and quickly create a new training session. Thus, GVT is now a full authoring platform to build virtual environments for training.

4 INDUSTRIAL VALIDATION: THE GVT PROJECT

The GVT project is developed in a Research/Industry collaboration, with three partners: INRIA and CERV laboratories, and Nexter group. This last partner is an important French company, specialized in military equipments such as the Leclerc tank. GVT is now an operational multi-user application with about 60 real industrial scenarios on 7 Nexter equipments. The main current application of GVT (the first release which is under commercialization) is virtual training on Nexter maintenance procedures, but GVT allows virtual training on more general procedures, such as starting procedures, showing procedures, diagnosis procedures.

In a typical training session, a trainer supervises several trainees (Figure 4). Each trainee, on his own computer, trains alone on one of the available procedures. The trainees and the trainer do not need to be co-located, they simply need to be on a common network. The paradigm of interaction in GVT is based on the selection of objects. A dynamic menu then appears so that the trainee can select the action to perform (Figure 1). This is a convenient choice to standardize the interaction. Furthermore GVT uses OpenMASK [2] for VR aspects. The use of this platform makes GVT totally independent from interaction devices which are managed by OpenMASK. Thus, different hardware configurations are available: from a fully immersive room to a laptop, including a dual-screen desktop computer (Figure 4). The peripherals used vary with the hardware configuration, but the software remains the same. In an immersive room, we can manage stereovision, speakers with voice synthesis (ex: to inform the trainee about the next action to perform), a microphone with voice recognition (ex: to change the viewpoint), a tracker which handles a pointer in the environment, and a dataglove to make the selection. On a laptop, a trainee can simply use the keyboard and the mouse to interact. For example, if speakers are not available, the voice synthesis is automatically replaced by text on the screen.

5 ADDITION OF COLLABORATION CAPABILITIES

The next generation of VETs is oriented towards collaborative aspects. Therefore, we have decided to enhance our platform, by adding collaboration opportunities for trainees [11]. In our state-of-the-art we identified two main limitations of existing collaborative VETs: a fixed distribution of scenario actions between team members and no parameterization possible for virtual humans' behavior. The models proposed tend to palliate these limitations.

We have implemented a model of the humanoid activity that suits both real users and virtual humans. A humanoid is a STORM object so that he can interact with other STORM objects, including other humanoids, depending on his abilities.

We have also made LORA evolve in order to allow the writing of collaborative procedures. For each scenario action, we now specify the roles allowed to perform it, and their priorities. Indeed, we have noticed that in some collaborative procedures, the sequence of actions to perform is more important than the person who performs them. We have thus decided to allow several roles for one scenario

[2] http://www.irisa.fr/bunraku/OpenMASK

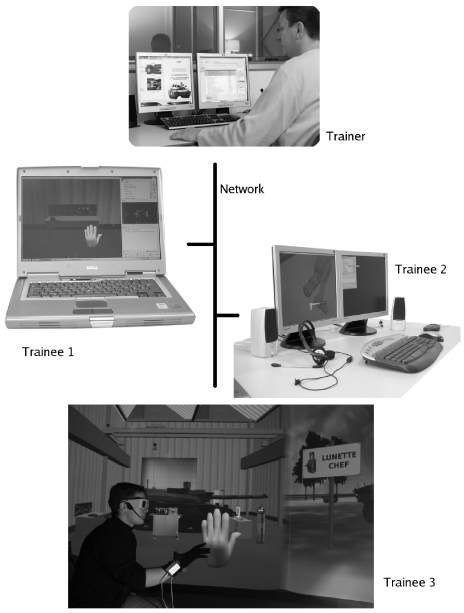

Figure 4: GVT on a network

action. This innovation leads to a flexible distribution of actions between actors while keeping the strict scheduling of these actions. The procedure is then more adaptive to a different context (number of people, location of each person and his availability, etc). Nevertheless, in return, it imposes to the set up a specific mechanism able to suggest who is the best candidate for each scenario action. We have also decided to simplify the writing of procedures, in order to improve the flexibility of scenarios and to increase productivity. Thus the writing of very common actions such as to take or to put back objects have been made implicit, which gets closer to the real specification of maintenance procedures. Instead, when needed, we specify the required tools in the action precondition.

In addition to these scenario modifications, we have developed a module of "action distribution", the aim of which is dual. The first one is to help a virtual human in his decision process because, for each action allowed in the scenario, this module suggests the best candidate from the scenario point of view. The second one is to propose pedagogical advices to a real user about the best action to perform. This module grades the different candidates for each scenario action according to their abilities to make the procedure progress. It can take into account various criteria such as the easiness for a humanoid to realize the action (the number of intermediate actions required), the proximity with the object to interact with, the adequacy of the role. Such a mechanism can be easily parameterized, by adding criteria or by giving a different weight to these criteria. The module of action distribution makes a global distribution of actions among humanoids, respecting the scenario requirements. But it is only a propositional distribution and it is up to each virtual human to make an individual choice thanks to his own decision module. Indeed, a virtual human is endowed with an individual decision-making process which chooses the next action to perform. It relies on the suggestion made by the module of action distribution and also on an individual collaborative profile (ex: individualist, helper, troublemaker). After gathering possible actions from various modules (the scenario via the "action distribution" mechanism, the pedagogy for special requests, the environment for all the possible actions (Figure 5)), it labels the actions depending on their types (actions belonging to the procedure, collaborative actions, hindering actions, etc). Finally, respecting as

much as possible selection rules that composed the collaborative profile of the virtual human, it selects the best action to realize.

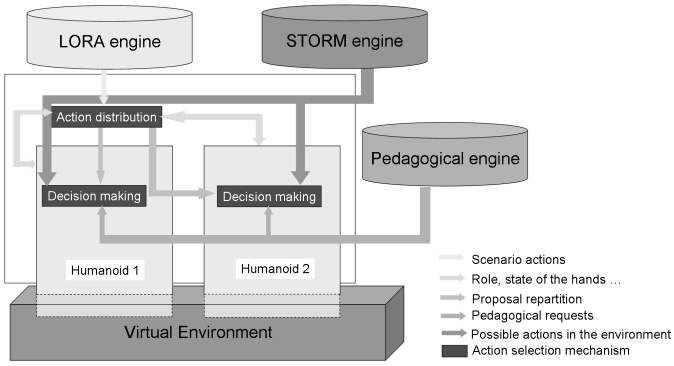

Figure 5: Action selection mechanism

The module of action distribution (centralized) combined with the decision modules of every virtual human (local) form the whole process of "action selection". This decomposition allows to have virtual humans with their own collaborative profile, but who are incited to follow the procedure and to make it progress in the best way. Moreover, this mechanism leads to interesting properties: increased flexibility of the distribution of the tasks, possibility to design or perfect procedures by observing virtual humans who realize the procedure, possible emergence of implicit collaboration such as a virtual human who helps a real user performing his task if he is blocked. Besides, STORM and LORA proved to be evolutionary enough to integrate both a humanoid model and collaboration opportunities.

The actual work consists in diversifying the available selection rules for the virtual humans. We also want to add collaborative actions to the scenario (when several actors must realize a common action, for example to carry together a heavy object) as well as communication actions (for example to give orders). At the present time, trainees and virtual humans must be located on the same computer. The next stage will be to distribute the application in order to allow distant real users to share a common virtual scene.

6 Assessment of procedural learning with the GVT environment

In order to be efficient, the conception and the assessment of a virtual training environment must rely on the knowledge of cognitive processes that occur in learning. Thus, in the domain of Cognitive Psychology, research about procedure learning related to the comprehension of instructions (see [10], in press) generally shows that in the first steps of learning (i.e., when the procedure is still unknown), the individual proceeds by the elaboration of one or several intermediate mental representations from the presented instructions in order to carry out the described actions, which is time and cognitive resources consuming (see [16],[5] and a review of different models [9](in press)). Then, procedural knowledge is considered acquired when the task, the material or the procedure are so familiar that it becomes possible to recover the procedure directly from memory [18]. That way, direct recovering of the procedure generally allows to reduce cognitive demands imposed by the conscious processing of the instructions [8][10]. Indeed, carrying out a procedure implies a more or less heavy cognitive load depending on whether it is achieved by simple recuperation of knowledge stored in long term memory or by the elaboration of intermediate mental representations (see [26] for a presentation of different levels of cognitive control in the acquisition of cognitive skills). In other words, when learning a procedure, one passes from a conscious to an automatic processing of the information [31, 30]. This passage needs a certain time and a certain cognitive effort (see Sweller's cognitive load theory [33]). Therefore, the goal of this study is to investigate how procedure learning takes place in a virtual environment. On the one hand, it aims at determining how many trials are necessary for the procedure to be considered as learnt, i.e., achieved in a very brief time, without reading the instructions and without error. On the other hand, it aims at verifying the acquisition of the procedure after a delay of more than six days in order to confirm, or to invalidate, its storage in an individual's long term memory.

This study, designed to investigate if procedural learning (i.e., the acquisition of procedural skills and their long term memorization) would take place using the GVT environment, is based on the theoretical framework proposed by Anderson [1, 2]. Anderson assumes that skill acquisition takes place in three stages. It begins during the cognitive stage with the use of declarative knowledge to guide new behavior. Here users typically work from instructions. They often represent the knowledge verbally and rehearse the instructions as they first perform the skill. Then during the second stage (the associative stage), the user makes a transition from a slow and deliberate use of the knowledge to a more direct representation of what to do: strengthening the connections among the various elements required for successful performance. At this stage users do not have to remember consciously - and to rehearse - how to proceed to perform the skill, and verbalization generally drops out. Errors that might have occurred in the initial understanding are gradually detected and eliminated. This leads to successful procedures for performing the skill. Finally, when these procedures have been repeated a certain number of times, they become more and more automated and rapid, and require few attentional resources. In this third stage (the autonomous stage) the acquired skill (stored in long term memory) is then revealed by errorless rapid performance.

This framework leads to the following assumptions: because of the systematic use of the instructions and the low knowledge of the environment, the task and the procedure, the first stage of skill acquisition would be time consuming and would lead to a high number of errors. In the second stage, time to read the instructions, to carry out the corresponding actions and number of errors would decrease till they reach the users' best performance in the third stage. At this level, if the skill is stored in a long term memory, performing the task after a one week delay would lead to a similar performance.

6.1 Method

In order to test these assumptions, learning with GVT was investigated using an instruction following paradigm. Twelve participants (10 men and 2 women ; mean age = 24) participated in the study. All of them were French engineer students (except one who was a young assistant professor) specialized in Computer-Science, but none of them had ever manipulated the GVT environment before.

After being showed a demonstration, the participants had to carry out a training task (a sixteen-step procedure) in order to become familiar with the environment. Then the experimental session began, where the participants had to follow textual instructions step by step on the computer to carry out a 25-step procedure. During this session, participants had to carry out the same procedure for ten trials. All participants were asked to come back to the laboratory one week later for three more trials (in order to assess whether the procedure was stored in long term memory). But only seven of them could come back to carry on the experiment.

The instructions and the equipment to manipulate were presented on separate screens (each masking the other one), so that reading time and execution time could be separately recorded. The participants could go from the instructions screen to the "Working Area" screen back again as often and for as long as they wished. The instructional materials used in the experiment were in French and developed from the original instructions provided by the manufacturer.

Table 1: Total time to perform the task, reading exposure time, execution time and number of errors

Trials	1	2	3	4	5	6	7	8	9	10	One week delay	11	12	13
Reading exposure time (sec.)	106.8	35.8	18.3	9.0	6.9	6.8	6.0	5.8	5.4	6.1		8.3	4.3	3.8
SD	33,53	16,62	11,11	7,99	6,05	3,44	3,95	4,28	4,38	4,38		5,52	3,26	3,58
Execution time (sec.)	326.9	174.9	135.3	121.8	119.1	111.8	109.2	108.0	106.6	126.1		126.7	107.5	109.9
SD	126,89	37,85	24,08	7,28	9,39	11,54	6,81	6,81	9,35	54,51		16,75	5,70	7,98
Total time (sec.)	433.8	210.8	153.7	130.8	126.0	118.6	115.2	113.8	112.0	132.2		135.0	111.8	113.7
SD	153,60	48,07	29,89	11,02	14,16	12,59	8,19	6,79	8,71	54,79		20,69	4,46	7,40
Number of errors	9,83	2,92	1,50	0,42	0,58	0,08	0,25	0,00	0,25	0,58		0,71	0,29	0,29
SD	10,12	4,42	2,11	0,90	1,00	0,29	0,45	0,00	0,45	1,44		1,11	0,49	0,76

Different kinds of measures were automatically recorded with the computer: reading exposure time (i.e., the time the instructional screen was displayed), execution time (i.e., the time taken to carry out the instructions in the working area screen) and the number of errors when performing the task. The reading exposure time data and the execution time data were analyzed separately and then put together to have an amount of total time to perform the task. Indeed, our assumptions were that the amount of total time to perform the task, the reading exposure time, the execution time, and the number of errors should decrease along with the repetition of the trials. After a delay greater than six days, the amount of total time to perform the task, as well as the reading exposure time, the execution time and the number of errors should be similar to those observed at the end of the first set of trials.

6.2 Results

Total time to perform the task was computed by adding reading exposure time and execution time. The results indicate significant differences between trials: F(9, 99)=42.06; $p < .001$; Mse=2798.18. Total time to perform the task decreases from trial 1 to trial 6 and then remains stable till trial 10, where a slight increase can be observed (probably due to a fatigue effect or a "last trial effect", due to the fact that the experimenter presented this trial as the last one) (Figure 6). Analytic comparisons show significant differences between trials 1 and 2, 2 and 3, 3 and 4, 4 and 5, 5 and 6, and no significant differences between other trials. After a one week delay, total time to perform the task slightly increases (trial 11) and then falls to reach the previously observed level of performance (trials 7, 8 and 9 vs 12 and 13: F(1,6)=1,46; ns). Exposure reading time, execution time and the number of errors follow the same pattern of results (Table 1). Note that reading exposure time never reaches zero sec., because it includes exposure time for the last instruction of the set ("Procedure is achieved").

6.3 Discussion

Using a simple instruction following paradigm, this study showed that a virtual training environment can help individuals to effectively learn a new procedure, i.e., store it in their long term memory so that it could be reused later. The results show that the total time to perform the task as well as the reading of the instructions, the execution time and the number of errors decrease along with the repetition of the trials. Moreover, after a one week break, these results are similar with those observed at the end of the first set of trials. These findings are congruent with the theoretical framework proposed by Anderson [1, 2] and confirm that the procedure seems to be acquired, that is stored in the learners' long term memory. Thus, this study seems to confirm that GVT, in an approach aiming at reproducing the real environment, brings to the learners the possibility to acquire a real knowledge of the procedure. One step

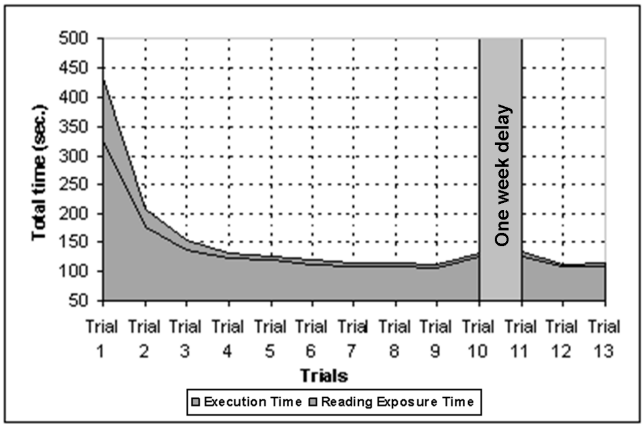

Figure 6: Means of total time to perform the task, reading exposure time and execution time

beyond would be to investigate whether procedural knowledge or cognitive skills acquired in such a VET could be transferred to the real world. To go further, some more research is needed to explore whether and how the design of such environments can be enhanced in order to improve information processing in the first steps of the procedural learning process.

7 Conclusion

Our goal is to facilitate the development of virtual training environments, and especially the reusability of previous developments for new projects. Therefore, we have developed a virtual reality training platform which is presented in this paper, based on generic models. The analysis of related works helped us to design STORM and LORA models, as well as a kernel based on these models supplemented by a pedagogical engine. The STORM model is a generalization of existing works and allows treating the virtual human as a standard object. It proposes a complete solution to design complex behavioral objects and complex interactions between such objects in a virtual environment. The LORA language, thanks to its graphical version, allows non computer scientists to author complex maintenance scenarios. This interpreted language is totally dynamic and allows authors of scenarios to create and modify the procedure in real-time, by using an innovative tool named "building by doing", one of our authoring tools for interactive prototyping.

Based on this kernel, the mature platform has grown and is now a full authoring platform to build virtual environments for procedural training. More than fifty operational scenarios have been de-

veloped on Nexter equipment. Furthermore, experiments are being conducted to evaluate and validate GVT, and a first assessment of procedural learning with GVT has been presented in this paper. Even though a first release of GVT is now being commercialized, research work keeps on going. The platform continues to evolve by integrating innovative models in order to offer cutting edge functionalities such as collaborative procedural learning. Indeed, a new prototype currently provides collaboration opportunities for the trainees, allowing them to train either with one another or with virtual humans.

ACKNOWLEDGEMENTS

The authors would like to thank all the members of the GVT project, specifically Frédéric Devillers and Xavier Larrodé, as well as Charlotte Hoareau for her generous assistance in collecting the data for the evaluation of GVT. The collaborative part of this work relies on the results of the Part@ge Project. Part@ge is granted by ANR (French National Agency for Research) under the reference 06 TOLOG 031

REFERENCES

[1] J. R. Anderson. *The Architecture of Cognition*. Harvard University Press, Cambridge, MA, USA, 1983.

[2] J. R. Anderson. *Learning and memory: An integrated approach*. New York: Wiley, 1995.

[3] N. I. Badler, C. A. Erignac, and Y. Liu. Virtual humans for validating maintenance procedures. *Commun. ACM*, 45(7):56–63, 2002.

[4] J. Cremer, J. Kearney, and Y. Papelis. Hcsm: a framework for behavior and scenario control in virtual environments. *ACM Trans. Model. Comput. Simul.*, 5(3):242–267, 1995.

[5] P. Dixon, K. Harrison, and D. Taylor. Effects of sentence form on the construction of mental plans from procedural discourse. *Can J Exp Psychol*, 47(2):375–400, June 1993.

[6] S. Donikian and B. Arnaldi. Complexity and concurrency for behavioral animation and simulation. In *Fifth Eurographics workshop on animation and simulation*, pages 101–113, 1994.

[7] J. Dugdale, B. Pavard, N. Pallamin, and M. E. Jed. Emergency fire incident training in a virtual world. In *Proceedings of the International workshop on Information Systems for Crisis Response and Management (ISCRAM 2004)*, May 2004.

[8] R. Engle, J. Carullo, and K. Collins. Individual differences in working memory for comprehension and following directions. *Journal of Educational Research*, 84(5):253–262, 1991.

[9] F. Ganier. Cognitive models of the processing of procedural instructions. in a. rothkegel & j. laffling (eds.) : Technology oriented communication. mouton de gruyter., in press.

[10] F. Ganier and J. Barcelina. *Considering users and their uses of procedural texts : a prerequisite for the design of appropriate documents*, pages 49–60. In D. Alamargot, P. Terrier & J.-M. Cellier (Eds.) : Improving the production and understanding of written documents in the workplace. Amsterdam : Elsevier, 2007.

[11] S. Gerbaud, N. Mollet, and B. Arnaldi. Virtual environments for training: from individual learning to collaboration with humanoids. In *Edutainment*, pages 116–127, Hong Kong, China, June 2007.

[12] M. Hildebrand, A. Eliëns, Z. Huang, and C. Visser. Interactive agents learning their environment. In *Intelligent virtual agents*, pages 13–17. Springer, 2003.

[13] T. Ishida. Q: A scenario description language for interactive agents. *Computer*, 35(11):42–47, 2002.

[14] W. L. Johnson and J. Rickel. Steve: an animated pedagogical agent for procedural training in virtual environments. *SIGART Bull.*, 8(1-4):16–21, 1997.

[15] M. Kallmann. *Object Interaction in Real-Time Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[16] D. E. Kieras and S. Bovair. The role of a mental model in learning to operate a device. pages 205–221, 1990.

[17] F. Lamarche and S. Donikian. Automatic orchestration of behaviours through the management of resources and priority levels. In *AAMAS'02 : Proceedings of Autonomous Agents and Multi Agent Systems*, Bologna, Italy, July 15-19 2002.

[18] J. Lefevre. Processing instructional texts and examples. *Canadian journal of psychology*, 41(3):351–364, 1987.

[19] M. Leitao, A. A. Sousa, and F. N. Ferreira. A scripting language for multi-level control of autonomous agents in a driving simulator. In *DSC'99*, pages 339–351, July 1999.

[20] N. Mollet and B. Arnaldi. Storytelling in virtual reality for training. In Z. Pan, R. Aylett, H. Diener, X. Jin, S. Göbel, and L. Li, editors, *Edutainment*, volume 3942 of *Lecture Notes in Computer Science*, pages 334–347. Springer, 2006.

[21] N. Mollet, S. Gerbaud, and B. Arnaldi. Storm: a generic interaction and behavioral model for 3d objects and humanoids in a virtual environment. In *Eurographics Symposium on Virtual Environments, Short Papers and Posters*, pages 95–100, 2007.

[22] J. Oliveira, M. Hosseini, S. Shirmohammadi, M. Cordea, E. Petriu, D. Petriu, and N. Georganas. Virtual theater for industrial training: A collaborative virtual environment. In *CSCC 2000: Proceedings of 4th WORLD MULTICONFERENCE on Circuits, Systems, Communications & Computers*, Greece, July 2000.

[23] M. Ponder, B. Herbelin, T. Molet, S. Schertenlieb, B. Ulicny, G. Papagiannakis, N. Magnenat-Thalmann, and D. Thalmann. Immersive vr decision training: telling interactive stories featuring advanced virtual human simulation technologies. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 97–106, New York, NY, USA, 2003. ACM Press.

[24] R. Querrec, C. Buche, E. Maffre, and P. Chevaillier. SécuRéVi : virtual environments for fire-fighting training. In S. Richir, P. Richard, and B. Taravel, editors, *5th virtual reality international conference (VRIC'03)*, pages 169–175, Laval, France, May 2003.

[25] R. Querrec and P. Chevaillier. Virtual storytelling for training : An application to fire-fighting in industrial environment. *International Conference on Virtual Storytelling ICVS 2001*, (Vol 2197):201–204, September 2001.

[26] J. Rasmussen. *Information processing and human-machine interaction*. Amsterdam: Elsevier, 1986.

[27] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[28] J. Rickel and W. L. Johnson. Virtual humans for team training in virtual reality. In *Proceedings of the Ninth International Conference on Artificial Intelligence in Education*, 1999.

[29] J. Rickel and W. L. Johnson. Extending virtual humans to support team training in virtual reality. *Exploring artificial intelligence in the new millennium*, pages 217–238, 2003.

[30] W. Schneider and R. M. Shiffrin. Controlled and automatic human information processing: I. detection, search, and attention. *Psychological Review*, 84(1):1–66, January 1977.

[31] R. M. Shiffrin and W. Schneider. Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory. *Psychological Review*, 84(2):127–190, March 1977.

[32] W. Swartout, J. Gratch, R. Hill, E. Hovy, S. Lindheim, J. Rickel, and D. Traum. Simulation meets hollywood: Integrating graphics, sound, story and character for immersive simulation. Multimodal Intelligent Information Presentation, 2005.

[33] J. Sweller. *Instructional design in technical area*. Victoria: Acer Press, 1999.

[34] D. Traum, J. Rickel, J. Gratch, and S. Marsella. Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 441–448, New York, NY, USA, 2003. ACM Press.

[35] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50, New York, NY, USA, 1994. ACM Press.

[36] M. van de Panne. Sensor-actuator networks. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press, 1993.