Entertainment Computing 5 (2014) 347–355

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Development of a mechanical maintenance training simulator in OpenSimulator for F-16 aircraft engines ☆

André Pinheiro [a], Paulo Fernandes [a], Ana Maia [a], Gonçalo Cruz [a], Daniela Pedrosa [b], Benjamim Fonseca [c], Hugo Paredes [c], Paulo Martins [c], Leonel Morgado [d,*], Jorge Rafael [e]

[a] Dep. Engenharias, Escola de Ciências e Tecnologia, UTAD – University of Trás-os-Montes e Alto Douro, 5001-801 Vila Real, Portugal
[b] Faculdade de Psicologia e de Ciências da Educação, UC – University of Coimbra, 3000 Coimbra, Portugal
[c] INESC TEC (Formerly INESC Porto), UTAD – University of Trás-os-Montes e Alto Douro, 5001-801 Vila Real, Portugal
[d] INESC TEC (Formerly INESC Porto), Universidade Aberta, Lisbon, Portugal
[e] Portuguese Air Force – Air Base Nr. 5, Serra de Porto de Urso, 2425-022 Monte Real, Portugal

ARTICLE INFO

Article history:
Received 31 May 2013
Revised 2 May 2014
Accepted 15 June 2014
Available online 22 June 2014

Keywords:
Virtual worlds
OpenSimulator
Virtual learning
Cooperation
Task coordination
Aircraft engine maintenance

ABSTRACT

Mechanical maintenance of F-16 engines is carried out as a team effort involving 3–4 skilled engine technicians, but the details of its procedures and requisites change constantly, to improve safety, optimize resources, and respond to knowledge learned from field outcomes. This provides a challenge for development of training simulators, since simulated actions risk becoming obsolete rapidly and require costly reimplementation. This paper presents the development of a 3D mechanical maintenance training simulator for this context, using a low-cost simulation platform and a software architecture that separates simulation control from simulation visualization, in view of enabling more agile adaptation of simulators. This specific simulator aims to enable technician training to be enhanced with cooperation and context prior to the training phase with actual physical engines. We provide data in support of the feasibility of this approach, describing the requirements that were identified with the Portuguese Air Force, the overall software architecture of the system, the current stage of the prototype, and the outcomes of the first field tests with users.

1. Introduction

At the Portuguese Air Force, engine technicians go through an initial training process at the Centre for Military and Technical Training of the Air Force (CFMTFA, Portuguese-language acronym), and are subsequently placed at different air bases, each with different aircraft, and specific engines and requirements. Thus, at each of these bases, they receive further training, focused on the specific engines and aircraft deployed and serviced there. In the case of the F-16 aircraft, this takes place at Air Base Nr. 5, in Serra do Porto de Urso, near Monte Real, Leiria, Portugal. Since technicians may be re-deployed to other bases, training of technical procedures for maintenance of specific engines is a common and frequent process. The training process has an initial theory phase, based on technical documents known as "Technical Orders" or TOs [1] (Fig. 1). Then an on-the-job training phase ensues, with trainees acting directly on an engine, in actual maintenance circumstances.

This final on-the-job training phase is resource-demanding, since it requires engines to be available either specifically for training or for longer servicing allowing for training to take place, and consequently unavailable for operation. Also, procedure errors, whose risk is greater during training, may in some cases produce costly component damage. Further, several of the technical procedures need to be executed by a team, meaning that time allocation of trainees, trainers, and experienced technicians needs to be managed, in order for a full team to be available for on-the-job training to ensue. These various resource requirements place constraints on the availability of on-the-job training opportunities and emphasize the need to optimize it. The development of a serious game for this scenario, a 3D multi-user mechanical training simulator, aims to provide trainees and trainers with more opportunities to conduct training, with the goal of allowing trainees to reach on-the-job training better prepared and thus to optimize the effectiveness of the resource-intensive training occasions with physical engines. This is a joint effort of the Portuguese Air Force and the University of Trás-os-Montes e Alto Douro (UTAD), which subsequently received the cooperation of the INESC TEC research organization.

☆ This paper has been recommended for acceptance by Bellotti.

* Corresponding author. Tel.: +351 222094000; fax: +351 222094050.
E-mail address: leonel.morgado@uab.pt (L. Morgado).

http://dx.doi.org/10.1016/j.entcom.2014.06.002
1875-9521/

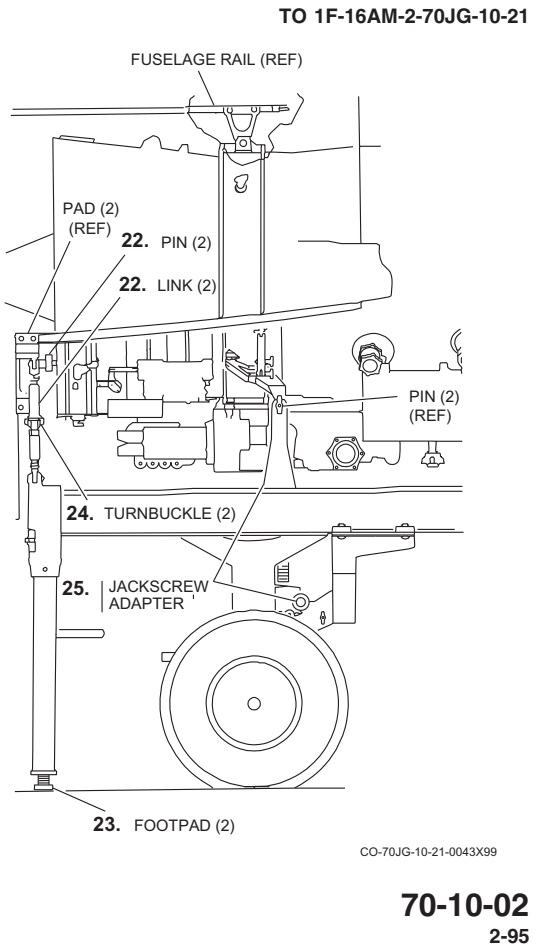

Fig. 1. Sample instructions from a technical order for the Pratt & Whitney F100-PW-220/220E engine from [1].

The development of such a serious game is a complex software engineering project, requiring technical expertise and a careful balance between design principles and pedagogical content, taking time, resources and teamwork. As serious games become more complex, so do the engineering challenges that arise during their development. A particularly significant challenge in this scenario is that mechanical procedures are constantly evolving: as errors are committed and/or insights developed, the recommended practice is changed, in order to optimize resources and lessen risks. At Air Base 5, this evolution effort is executed under the approach known as lean principles [19].

The consequence for software development is that any mechanical procedures implemented in a simulator are likely to become obsolete rather quickly. Any development costs and resources are further increased by the need to update the simulated procedures. This is now somewhat lessened by employing game engines or development platforms rather than developing from scratch, rendering the early-stage selection of the engine or platform for development critical [2]. Following this software engineering perspective, there is a goal of lessening the resource requirements of both simulation development and updating. Thus we present an approach that focus on tuning and perfecting the simulated operations and behaviors, rather than the visuals. For this purpose, we employed a readily available virtual world platform (OpenSimulator) rather than a game engine; and in order to allow the knowledge embedded in behaviors and operations to be independent from this platform, we implemented control code and decision-making logic as an external system, accessed as a Web Service. The rationale for this architectural choice was to enable the visual and interaction platform to change subsequently, if necessary, but keeping the fast-prototyping benefits of a virtual world platform such as OpenSimulator [3]. In this paper, we present this approach and test its feasibility by submitting the prototype to user tests at the air base, with mechanical maintenance trainers.

2. Background

F-16 aircraft of the Portuguese Air Force are at the so-called mid-life update version (known as MLU), and employ Pratt & Whitney F100-PW-220/220E engines, with large number of mechanical maintenance procedures – the manufacturer recommends periodical inspections depending on the number of flight hours. Inspection procedures are conducted before and after each flight, and there is also programmed maintenance that takes place every 300 flight hours, with the overwhelming majority of these procedures taking place at air base Nr. 5. We held meetings with the training team and mechanical experts at air base Nr. 5, to ascertain the most relevant procedures for technicians that are initiating their training with this specific engine. In the course of those meetings, the procedures for installation of the engine inside the F-16 aircraft fuselage were selected as the first simulation target. This involves a series of steps for properly installing and connecting the engine, which need to be done not only effectively but also safely. We have collected data on this process by combining several sources: we reviewed the TOs [1] and taped and photographed the actual current installation process for the engine from various perspectives. We then decoded this data, describing it in terms of a natural-language script, and created UML diagrams of the various procedures, keeping in regular contact with Air Force trainers to clear out doubts and get further details. Briefly, these steps involve preparing the engine for transportation towards the aircraft fuselage, transporting it and preparing it for insertion, raising it and inserting it into the aircraft fuselage, establishing the engine connections to the fuselage, and testing the installation (the full list is provided in Table 1). The level of detail required for simulation of each task was also determined in cooperation with the trainers at Air Base Nr. 5.

For creating the simulation, given that this project was developed with minimal funding, it was be necessary that development could be incremental, in small steps over time, likely involving different people in each academic year. In the meetings with the training team at the air base, context-specific requirements were

Table 1
Procedures involved in the installation of the engine into the aircraft.

Nr.	Task description
1	Installation of the engine mount to raise the engine
2	Engine raise
3	Couple engine to the aircraft
4	Perfect alignment of the upper engine mount with the fuselage rail
5	Fasten and break of the trailer to the fuselage
6	Adjustment of the trailer to transfer engine weight
7	Support the engine on the fuselage rail
8	Enter the engine in perfect alignment with the fuselage
9	Align with the trust pin connections
10	Enter the trust pins using the connection doors
11	Finish the back
12	Check perfect alignment of the trust pin with the clamp half
13	Tighten up the clamp half with a ring (nut) to finish baking
14	Finish installing the trust pin with an inspection
15	Check the SEAL, from the air entrance
16	Remove the trailer
17	Remove the back engine adapter
18	Lighten/download support fuselage
19	Completely remove the trailer

established. First, the main usage focus is for trainees in a training room at the air base, but with flexibility for later use from their homes, or with the participation of trainees at other military locations. Further, as mentioned earlier, the actual mechanical procedures might change, requiring regular simulation updating. This led us to consider virtual worlds as a development platform, to lessen the development requirements of the underlying simulation environment, benefiting from pre-existing networking and multi-user features of these platforms. The rationale was that virtual world platforms provide a set of basic features such as content rendering, user login and interaction, user messaging, and object physics, among other aspects. This enabled development to focus on the behavioral elements of simulation development: the core knowledge of the simulation. In order to leverage the scenario for this purpose, we set to plan the prototype development using OpenSimulator, a virtual world platform with the ability to enable communication with external systems via scripts, thus without requiring changes to the underlying code of the virtual world platform. OpenSimulator employs the client–server protocol of Second Life and therefore we could also benefit from the full set of coding resources and script-developing communities that exist for both platforms. OpenSimulator specifically is used by many different groups and for different purposes, and as with many virtual worlds it supports collaboration, including awareness of the presence of other users and communication, immersive interaction, and a credible 3D representation. The role of OpenSimulator in the simulation community has been defended for various scenarios, excluding pure science and target user groups of simulation experts [12]. We have also been exploiting the development of this scenario to create and further a simulation control architecture which attempts to have more concern independence between the simulation's behavior logic and its human interaction and visual aspects [3].

The use of virtual world platforms as the environment for developing and deploying training simulators is frequent, in fields as diverse as emergency response [4], business management [5], medical and health scenarios [6], and security forces [7]. Military training scenarios have also employed virtual worlds as their development platform, such as the OLIVE platform [8], and this use follows in the stead of a long history of gaming and gaming technologies in military training [9]. Training is critical for the success of military operations, including background technical operations such as aircraft maintenance, not just for tactical operations and combat. In this sense, virtual world environments with multi-user abilities allow personnel to interact in a simulated face-to-face environment. There are several examples of serious games used by multiple users for training combat and tactical activities [10,11] but the field of multi-user technical training simulations is still in the beginning. Some examples exist, such as the work done at the University of Pennsylvania [14], the main differences for this work being that we focus on multiple coordinating individuals rather than a single user.

The field of serious games, of which this work is part of, has been receiving increasing public interest and awareness, emerging as a sort of accepted terminology. Many well-surveyed publications in the emerging serious games literature highlight the often conflicting interpretations of serious games in an attempt to pin-down what the term actually encapsulates [16]. It challenges our understanding of characteristics such as, challenge, play and fun, which are fully associated with video games. However, they may not be appropriate for all serious games. For some serious games, it's more fitting to talk about experience rather than fun or entertainment, like the training simulator presented in this paper. Characteristics such as being thought-provoking, informative, awareness-raising or stimulating are as important, if not more so, than being fun or entertaining. In this sense, we adopted Marsh's definition [16], which considers serious games along a continuum that can be framed between games with a specific purpose at one end, and environments for experiencing, with little or no gaming features at the other end.

We consider this continuum to classify our simulator as a serious game with reduced gaming characteristics and serious experiential and cultural purposes. Other examples in this area of the continuum are the many purpose-enabled simulations created in the Second Life virtual world, because of their reduced gaming characteristics, mechanics and gameplay: their purpose is to provide potentials or opportunities for experience and emotion through encounters to provide meaning. It also includes the class of training simulators.

According to Narayanasamy et al. [17], training simulators present specific characteristics, some of which are similar to games and/or simulation games (1 & 2 of the following), but others are unique (all the others): 1 – presence of a virtual environment; 2 – interactive user engagement in a form of simulation; 3 – restricted to providing recreations of real-world environments; 4 – no intention of entertaining, amusing, or engaging; 5 – user may happen to find the application entertaining, fun, or engaging; 6 – the primary purpose is the development of user skills; 7 – accurate depiction of challenges regarding the real-world scenario; 8 – focus on standard operational procedures, instead of gameplay patterns or gestalt; 9 – stable procedures; 10 – activities are not goal-oriented; and 11 – the end-state is not obvious.

The reevaluation of simulator approaches to incorporate game and game-like elements places an increasing demand for serious game developers to deliver high-fidelity solutions. Some authors [2] proposed a game engine selection framework for serious applications development that includes elements as fidelity, heterogeneity, composability, consistency, accessibility and networking: regarding fidelity, they point out some factors such as narrative, depth of visual/auditory content, interaction medium and character/object behavior; for heterogeneity, they refer to the diversity of platforms on which the engine be deployed, including hardware requirements and scalability; for composability, they point to the features allowing game engine content to be reused, both that created within the game engine, and that imported from other sources; for consistency, they point out several technical features, including whether game areas need to be loaded, if the engine is for standalone games or Web-based games, and more; for accessibility, they point out support for non-standard interfaces and devices and the extent of as support for standard interfaces, but also to the engine's support of both expert and novice developers; finally, for networking, they point to the issue of multiuser support. This last one is particularly relevant in training, since human instructors have a significant role in the outcome and learning impact of simulator-based learning [20].

3. System architecture

As mentioned earlier, a key concern in this system is that its operations may need to be changed regularly: while the TOs are the basic reference for maintenance procedures, their actual execution is tuned to improve efficiency or diminish risks, following a perspective of continuous improvement and lean principles [19]. Further, in order to include gaming approaches to training, different pathways, error possibilities, and varying degrees of difficulty may need to be implemented and subsequently redesigned in response to changes in the established procedures. Therefore, we strived to maintain independence between the visual elements of the simulator and its state control and decision-making. For this purpose, we took advantage of the fact that OpenSimulator virtual worlds can execute scripts concurrently, with each script able to

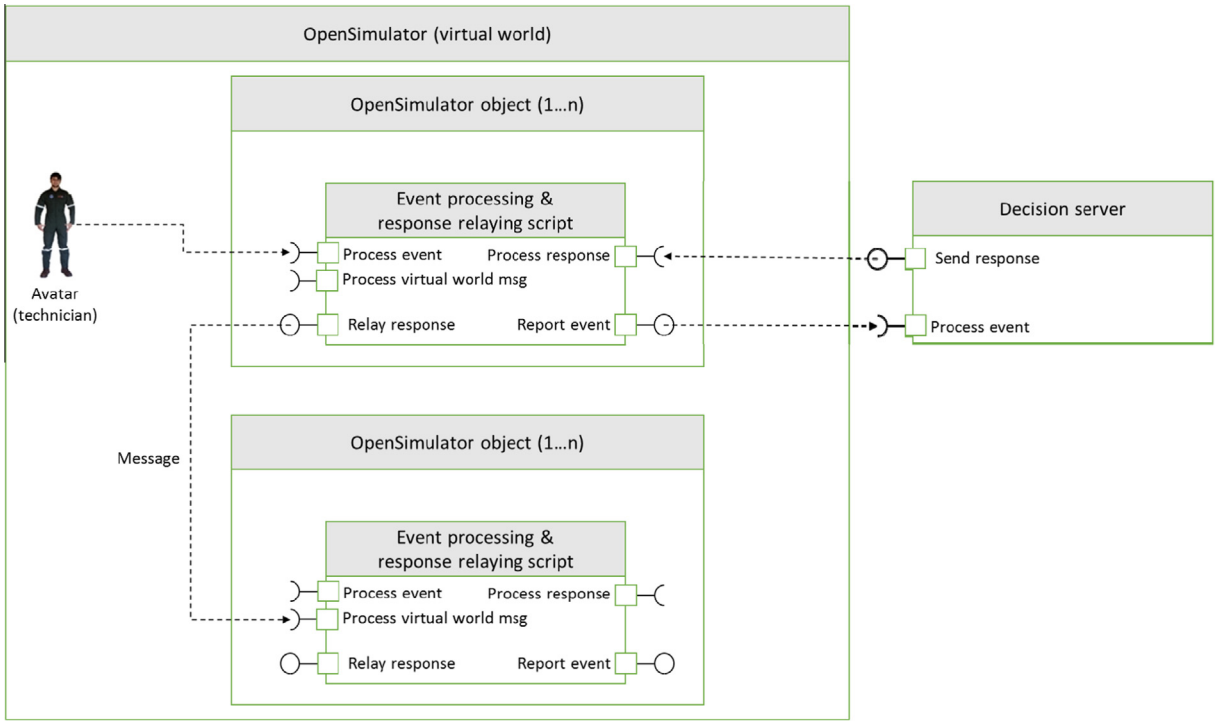

Fig. 2. System architecture (adapted and clarified from [3]).

Table 2
Protocol packet format used between the Web Service and OpenSimulator scripts.

Packet structure	<number of commands><endline><command 1><endline><command 2><endline>...<command n>	
Element	Type	Description
Number of commands	Integer number	Number of commands included in the list
Command	List	<task>,<nr.parameters> <param. 1>,<param. 2>,...,<param. n> Where task = integer number, identifies the task: move, rotate, create object, etc. nr. parameters = integer number, indicates how many parameters are provided in this task param = comma-separated strings, each containing the data for a parameter

communicate with external software servers, and developed the architecture presented in Fig. 2.

Therefore, whenever a technician interacts with the system he/she controls an avatar inside the virtual world (as shown in Fig. 2). The actions of these avatars upon virtual world objects generate events, which are detected by the objects' scripts normally. Instead of having the simulator actions hard-coded into these scripts, the scripts only report any events to an external decision server system for overall state control and decision-making (this is done using OpenSimulator's HTTP request services). This control and decision-making includes checking for cooperation and synchronization requirements (i.e., tasks that need to be performed in concert by several technicians). The decision server then responds to the originating script with a list of actions to be performed by the virtual world objects. Upon receiving this response, the script reacts, causing the visual appearance of the simulation to change. This is done by each object's own script: the script receiving the response from the decision server acts upon its host object directly and relays to scripts in other objects actions meant for them, using the internal messaging services of the virtual world. This process provides the technicians (via their avatars) a new state for their intervention.

A central aspect to this architecture in OpenSimulator is that for some actions a single script may not be able to originate the entire change of state required in the visual simulation. For instance, since scripts are linked to specific objects, any decision that involves two or more objects needs to involve an identical number of scripts, not just the originating script. For this reason, when the

Table 3
Protocol packet format used between scripts in OpenSimulator objects to relay commands.

Packet structure	<channel>,<key>,<task>,<nr.parameters>,<param. 1>,<param. 2>,...,<param. n>	
Element	Type	Description
Channel	Integer number	Communication channel being used
Key	String	Identifier of the script to which the task is destined
Nr. parameters	Integer number	Indicates how many parameters are provided in this task
Param 1, 2... n	Strings	Each of these contains the data for a parameter

decision server needs to impact several scripts, the associated commands are included in a list which is sent as the response to the script that originally reported a technician's action. That script is in charge of relaying the commands in that list to the other scripts in the simulation, achieving the intended multi-object result.

The adopted solution was to create messaging protocols for transmission of command from the decision server to the reporting script (Web Service–OpenSim protocol, see Table 2) and for relaying of commands between the reporting script and the other scripts in the simulator (OpenSim–OpenSim protocol, see Table 3).

Since all scripts behave identically (detecting avatar actions, reporting them, receiving lists of commands, relaying command, and executing them), this architecture in effect assumes that all concurrent scripts are in effect copies of a single piece of code, and each script employs the identifier of its hosting object as its own. This means that all simulation-specific and game-specific implementation details are externally-hosted components, and thus any changes in mechanical procedures will impact the external components, but not the visual game code or virtual world scripts themselves. Even creation of new objects within a game or simulator is simplified, since those objects simply get their own identifiers and a copy of the same piece of scripting code. Further, debugging and updating processes are streamlined: since the same code is used in all scripts, it simply is updated everywhere, rather than having to identify and implement changes separately for different game or simulation elements.

We emphasize that the 3D environment does not conduct any decision-making: it is only responsible for reporting to the back-office web service at the decision server the events triggered by the technician's avatars or other objects and then waiting for the commands to be issued by the Web Service, in response to these reports. In this sense, it is an embryonic implementation of the Model-View-Controller architectural style [13] for virtual worlds. This holds the potential to render decision-making independent from the 3D environment, not just for this system but as a generic approach.

4. Prototype aspects

To develop the system prototype, we expedited the 3D environment modeling using the built-in end-user tools of OpenSimulator/Second Life client viewers, and employed QAvimator to recreate in 3D the movements of the technicians. The goal in this prototype stage was not one of photorealistic visuals, but simply to be credible for testing and development. Following the overall goal of separation of concerns between simulation rendering and software control/decision-making specified in the architecture, we implemented control and decision-making as an autonomous system, available to the OpenSimulator platform as a Web Service. The decision-making was implemented in the form of a hierarchical state machine.

In order to separate the control system from the specificities of OpenSimulator client–server communication, we implemented the protocols specified in the architecture using plain text comma-separated values (for performance reasons), and a single piece of code to be used in all scripts within OpenSimulator. This piece of code was developed in LSL (Linden Scripting Language, commonly used for scripting in Second Life and OpenSimulator worlds).

Albeit this prototype is still in an early form, we were able to confirm that already it fulfils the intended architectural goal of separation of visuals and control. We can replace the decision-making algorithm entirely, without having to change a single line of code in the 3D environment: in a parallel effort, we have used the same 3D models and scenario to implement the Partial-Order-Planning algorithm as a reasoning model to replace some of the human-controlled avatars by an intelligent software agent [18], and the approach we used enables us to now combine these decision-making approaches without having to change the 3D scripts.

Currently, the 3D script is the one aspect which still binds the system to a specific virtual world technology. It is reporting OpenSimulator/Second Life events and its copies are meant to be running concurrently. But the script is detached from the decision-making process. Thus, we have the ambition of making this approach evolve in order to render it viable for other virtual world platforms and game engines. Ideally, by developing a new event-reporting module and a new translation mechanism of decision-making commands into the specific requirements of the each technological platform, one would like to be able to benefit from updating the visual technology and platforms of existing simulations with much lessened development efforts and resources.

The current decision-making system, while not the focus of our development concerns, is presented herein for clarity. It works as follows: when it receives an event from OpenSimulator, it first queries a data store with all current data on simulation state. Then, based on this information and the date of the OpenSimulator-originating event, a hierarchical state machine algorithm determines the adequate response. For example, if a technician is in a "Free Hands" state, this state can transition to the state named "HoldingScrewdriverInHand" upon receiving a "ClickedOnScrewdriver" event. Finally, the transition is translated into virtual-world specific commands, which are provided to the originating script for execution.

5. Sample simulator tasks

The installation process of a Pratt & Whitney F100 engine in an F-16 aircraft is quite extensive and complex, requiring three technicians to do the various procedures. Plus, a specific role in the process is that of process checker, which may be played by one of the three operating technicians, by may also lead to a fourth person being involved, should none of the three required technicians have the credentials to perform this role. All necessary procedures are specified in a document known as the Job Guide, which refers to all the Technical Orders containing the necessary information. Installation of the engine inside the aircraft fuselage, the first procedure being implemented in the prototype, is a single process, but it is typically subdivided into four jobs, known as PT1, PT2, PT3, and PT4. Currently the prototype is implemented to support simulation throughout the PT1 job, comprising tasks 1–8 of Table 1. This job is a precondition for executing jobs PT2, PT3, and PT4, which comprising the remaining tasks in Table 1. But more importantly,

Fig. 3. Mechanics raising the engine at Air Base Nr. 5.

this prototype enabled testing of the cooperation situations of the process, because this PT1 job requires the involvement of all technicians, as it entails several tasks that cannot be performed by a single individual.

One such task is the raising of the engine to align it with the empty hull of the aircraft fuselage, which requires all 3 technicians: two on the left side, another on the right side, as shown in Fig. 3. When the process checker issues the command (i.e., either one of the three technicians or the fourth element in the team), the three technicians will operate screw-driving machines in concert to lift the motor simultaneously and coordinately, using speech as a means of synchronization, to avoid tilting the engine excessively while raising it.

The multi-user virtual world platform OpenSimulator enables three or four trainees to practice the synchronization in this task in a similar way, as long as voice chatting is available (this can either be enabled as a free OpenSimulator module or as a parallel voice chatting application). In the current prototype, since the main focus is in support of training with all trainees in the same physical room, we have not yet installed voice chatting features. The users in the simulation proceed with the operations, and once all preconditions are adequate (such as having steadied the bearing cart and having adequate tools in hand and fitting for the tools in place), the engine lifting can take place. Once it is initiated, we elected to implement it in the following way: the simulator will take control of the arrow keys on the keyboard and change their function (usually, they move the avatar). During the lifting process, up and down arrow keys will respond as if operating the direction of screw driving machines or the turning of lifting wrenches (Fig. 4a).

The three technicians need to press their keys in the correct direction at roughly the same time (using voice chatting to synchronize their actions) in order to lift or lower the engine. We defined "roughly" as a 2-s interval, but could make it stricter. Currently, this is a precondition for lifting/lowering to occur, but in the future we intend to expand the simulator behavior to cover error conditions and accidents caused by wrong operations. Fig. 4b shows the avatars of trainees during the lifting of the engine in the virtual space.

6. User testing

6.1. Settings, preparation, context and testing

In order to evaluate the feasibility of the simulator prototype, to ascertain whether we could pursue this architectural approach and inform its subsequent development, two field tests with prospective users were planned and conducted. Test 1 took place during a 2-day stay at Air Base Nr. 5. An early version of the prototype had been demonstrated earlier to the air base command, but no actual user tests had been conducted then. With this 2-day stay, the team aimed to make the prototype available to technicians that are also trainers, in order to identify their satisfaction as users, and their expectations regarding subsequent development – particularly in areas such as technical details and pedagogical affordances, user interface choices, and interaction methods. We also wanted to ascertain whether the technological context at the air base (computer hardware and network) supported the execution requirements of the prototype.

Fig. 5. Room arrangement during Test 1.

Since the air base is located 260 km away from UTAD, where most of the development team was located, and funding was minimal, the preparation for Test 1 was done remotely, via e-mail exchanges. Two test sessions were planned, which included 3 trainers each, chosen on the basis of being potential future users of the simulator in the context of training sessions. These trainers were the people in charge of training new technicians as they are assigned to the air base, for operation on F-16 aircraft engines. I.e., they were the air base most qualified technicians regarding the mechanical maintenance procedures for F-16 engines. The 2 sessions were similar. Both took place in a side room of the main engine maintenance hangar (Fig. 5). This room had 2 group tables with chairs, a projector and a white board. 4 laptop computers were used for tests, 3 for the users and 1 operating as an OpenSimulator server. The test preparation consisted in gathering and assembling all necessary supporting materials (computers, networking, cameras and microphones for test recording, software installation, and distribution of consent forms for data collection, questionnaires for characterization of user profiles, and paper guide).

The first day of our stay was employed in room preparation and installation of the software on the air base computers. The actual training sessions took place on day 2, each lasting about 1 h 15 min.

The sessions consisted of a brief introduction with a project presentation, an explanation of the motive for the team being present,

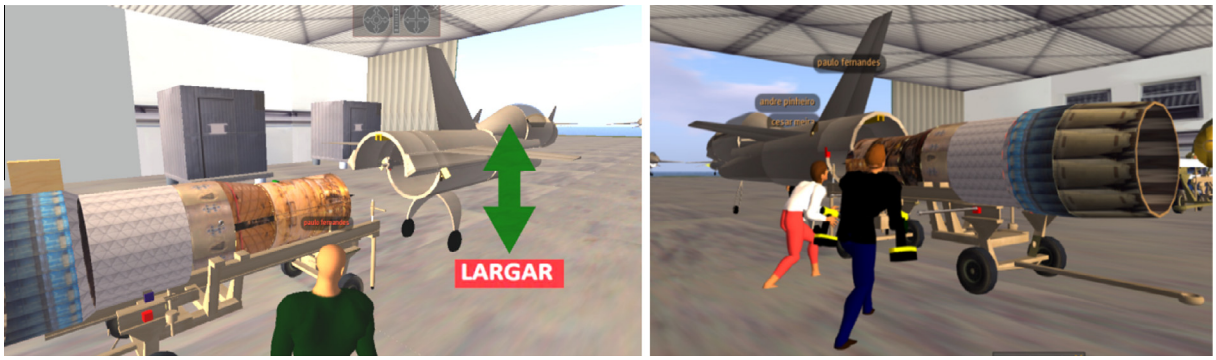

Fig. 4. Simulator aspects: (a) arrow in virtual space and "Largar" (drop) button to drop the screw-driving machine; (b) mechanics lifting the engine in the virtual space.

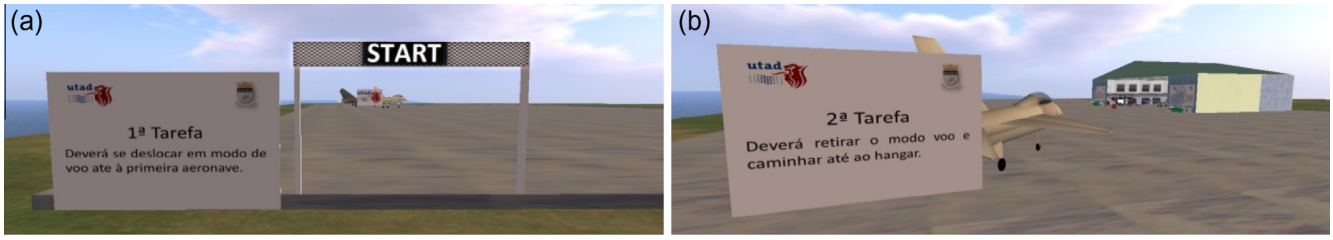

Fig. 6. Setting activity: (a) first station; (b) second station.

Fig. 7. Setting activity: third and fourth stations.

and a presentation of the work plan for the session. All trainers read and signed consent forms regarding data collection for this test, and filled in a characterization questionnaire. During the simulation session proper, the whole dynamic was being projected onto a screen, from the perspective of the computer whose user (a development team member) played the role of supervisor for the engine installation simulation. The session began with an activity for acquainting users with the platform, aimed at the acquisition of basic skills for moving in the virtual world and interacting with objects.

This acquaintance activity had 4 steps. The group of avatars started by meeting in a location away from the virtual hangar. At each step there was a virtual poster with a task for users to accomplish. The first was to enter avatar flight mode and move near the aircraft parked outside the hangar (Fig. 6a). Then they had to walk into the hangar, up to a table with objects on it (Fig. 6b). At this table they had to pick the objects, saving them to their avatars' inventories. Next, we asked them to carry the objects to another table, near the first one (Fig. 7). Here they had to place the objects on this second table, i.e., taking them from the inventory. Finally, the acquaintance activity was completed by asking them to return to the aircraft parking location, and from there to the engine, to initiate the simulation of engine installation.

During the engine maintenance simulation, trainers were asked to employ the think aloud protocol throughout the process, communicating their thoughts and feelings. After completing the simulation process, a final group interview took place, recorded for later analysis. The interview attempted to measure users' satisfaction towards the system, and collect suggestions and recommendations for improvement.

6.2. Questionnaire results: trainers' profile

By analyzing the questionnaires, we ascertained the following: all trainers taking part in this test were male, with an average age of 42. They reported being regular computer users, using computers more than once per day. Their main uses of computing technology are searching for information (100%), work completion (83%), and socialization (83%). Regarding the use of virtual worlds, none of the trainers had used them before, nor were they aware of them.

6.3. Problems and limitations found

6.3.1. With the local hardware

The air base had assigned for this test: four laptops, a network switch, and network cables. We found out that this equipment was not entirely suitable: to run this prototype, we needed a network gateway, a service that was not provided by the provided network switch. So we proceeded with testing by employing a small router we had brought along as an alternative for such a case. The laptops had adequate memory and processing powers, but some limitations in terms of graphic cards, which were able to run the client software (Second Life viewer) but not in the best conditions. One was replaced by a developer's laptop so that the test could take place.

The computer being used as server also revealed itself as providing insufficient performance, and this exposed some deficiencies in data consistency between the Web Service and the 3D environment.

6.3.2. With the simulator prototype

From analysis of interviews, respondents (trainers) mentioned some aspects for improvement and correction. For instance, avatar positioning had not been identified as a requirement earlier, but upon observation by experts it was noticed that technicians were not always placing their virtual bodies correctly during procedures. This is an important component of training, since inadequate positioning may expose a technician to unnecessary danger or inability to perform adequately, and will therefore have to be considered for future prototypes.

Regarding the usefulness of the simulator, respondents found it useful, an asset in support of practice. But they expressed the need to take into consideration not only the actual procedures, but also the prior safety inspection checklist. For instance, they used expressions such as: "In terms of safety of the airplane stability, this is why I told you of the amount of fuel that..." (a plane without enough fuel weight will not remain stable on the hangar floor if the engine is removed); "That part is always important (...) to safeguard his safety of the rest of the team's work"; "While typically the aircraft will have all these elements, sometimes (...) a seat may not be there (...) which needs to be corrected..." (Respondent 2, March 2nd, 2012).

6.3.3. Other aspects mentioned in the interviews

From a user perspective, respondents of both groups classified the level of knowledge and preparation required to deploy the 3D virtual world as acceptable, although depending on actual computer and software capabilities.

Regarding the acquaintance activity with the virtual world, respondents felt it facilitated the contact, with expressions such as "Ah ... Yes, yes", "[Contributes] to knowing the steps...", "It is an introduction" (Respondent 1, March 2nd, 2012). They found

the time for the acquaintance activity to be "enough" and "appropriate" (Respondent 1, *id.*).

Regarding the use of the actual simulator, the usefulness of having an introduction was reported, as was the importance of a checklist with common errors and required tasks to accomplish in the simulator. Trainers considered that support by training staff and colleagues will be important at an early phase, but will eventually cease to be required. E.g., using statements such as "the second time I think I got it..." (Respondent 1, *id.*).

As for the pedagogical content of the simulator, trainers considered that it did allow training of necessary skills for installation of F-16 engines, but noted that it is necessary to increase the level of detail. E.g., with statements such as literally "more details" or "Instead of placing things on the ground, to check materials, you could watch them here..." (Respondent 1, *id.*). On the collaboration aspects of simulator use, they considered that it achieved the desired goals, since it allowed the practice of these aspects.

Overall, they found that that the simulator could benefit the training of the installation of F-16 engines, reporting aspects such as "the advantage of the person being able to correct mistakes" (Respondent 2, *id.*), and "when new staff arrives, they learn, they do stuff, but until a team gets automated it takes time" (Respondent 2, *id.*).

On the subject of virtual worlds, they considered them an alternative to the physical training of tasks, and an appealing one, able to "create more enthusiasm" (Respondent 2, *id.*).

Regarding the graphical interface, trainers pointed out that some engine parts have to be more realistic, to the point of being able to identify the part when seen "really close" (Respondent 2, *id.*). They did not find it difficult to identify the objects, but corrections are needed, since some aspects simply do not occur in reality (e.g., "the engine cannot leave the front, that's impossible" – Respondent 2, *id.*).

They pointed out that interaction with the system is adequate and sufficient, yet that there is the need to get used to the conditions that the system offers.

Finally, some suggestions were provided regarding technical training, including on the issue of safety awareness, and details such as ensuring the use of a toolbox in certain circumstances, or identification of occasional errors in the installation sequence of tasks. Interface cueing requests such as displaying an object description or name upon it on mouse over, were also recorded.

7. Refinement testing

7.1. Settings and conducting testing

After improving the simulator, based on the results of Test 1, we conducted a new field test with prospective users, on July 2nd. The setup, preparation, and physical context was identical to the first test, but this time with the simpler goal of confirming whether new procedures and modified procedures were being provided correctly. Again two sessions took place, involving six trainers, divided into 3-element groups (since 3 technicians is the minimal number for conducting the engine-insertion procedure). The members of each group were on tables this time placed in a square configuration, in order for them to be unable to see each other's screens (Fig. 8).

This test was initiated with a brief project presentation, and subsequently the virtual world introduction activity previously mentioned, since not all participants had been part of the original test. Afterwards, the full mechanical maintenance process was simulated. A subsequent semi-structured interview aimed to gather new data on trainers' satisfaction and collect improvement suggestions.

Fig. 8. Room arrangement during Test 2.

7.2. Results

During this second test, further changes required to the sequence of operations were detected and noted for later correction. Some were misunderstandings of the development team, others had changed during the development process. As explained earlier, the procedures registered in the TOs are changed regularly as part of constant improvement and lean maintenance principles. Sample feedback about this: "Now it's like this...does this allow the engine to be totally inserted in this step? There is a procedure missing... yes, extremities need to be pulled back"; "The seventh one needs to be pulled back...someone needs to deviate it while it goes in"; "we need to lower the entire support that the cart provides to the front (...) when it is fit we lower the cart support".

Computer skills to use the simulator were once more considered accessible, without major demands from expected end users – engine technicians. Sample sentences attesting this: "...it's a matter of simple touches: we want a tool, and just click it. It has some parts in English, but they're basic, like «fly»"; "it's just a matter of pressing and it's done."

The graphics of some parts of the simulation were pointed out as a possible deterrent from action, and in the final environment more realism is needed: "...that red dot, for first-timers, isn't clear... it doesn't tell whether if it is meant to lift or remove the cart."

The simulator was again seen as useful for training, in particular for new technicians that may join the team: "Yes, for those arriving into this section, initiating, it's cool"; "...one gets a notion, then completes it...instead of being in the field and then saying «now reach me that part» and then one cannot see what's taking place at the front, or one is unable to see what the ones at the plane sides are doing. One doesn't get a clear notion that everything must be done at the same time..."; "they would have a different perspective regarding their place."

More operational requirements were set forth. E.g.: "if this had some variation to match slope, it would be more noticeable... in here we can witness all the steps, while in the field we can't..."

8. Conclusions and future work

The tests confirmed that the selection of a virtual world platform technology (OpenSimulator) and the use of a Web service for control of the simulation is a feasible approach for the development of training simulators for this scenario, considering the use by the actual trainers involved in technician training at the air base. The nature of most omissions in terms of simulation aspects revealed that the procedure analysis needs to be fine-tuned, taking into account tactical know-how and systemic aspects (such as the security component of certain avatar positions or the importance of using specific accessories such as the toolbox for better team

coordination), but did not stem from platform or architectural shortcomings. One possible bottleneck is that while the overall graphics quality is adequate, specific parts need higher quality for better association between them and their intended purpose. OpenSimulator enables much better visuals than the ones we used, but the level of control of the user experience (e.g., automated camera control and automated user avatar control) is lesser and chunkier than that available in game engine platforms, so further exploration of this risk is needed. Expanding the simulator to support mixed teams of human-controlled avatars and artificial intelligence agents, to support training even if only some of the human trainees are available, is another promising line of work, which other teams have pursued [15], and which we have also started to explore [18].

A promising aspect was the relevance of the adopted strategy of separating implementation from visuals: even in the relatively short time involved with the development of this prototype, some procedures changed; due to independence from the rendering platform, such changes have a higher likelihood of requiring less development resources than if they implied both control and visual edits.

The prototype is being further developed and the architectural goals of independence between virtual world platform and decision-making/control logic have shown their feasibility. We hope not only to complete the simulator, but to be able to pursue further the development of this software engineering approach, in view of faster and less resource-consuming serious game development.

Acknowledgements

This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP – 01-0124-FEDER-022701». This project has been partially funded with support from the European Commission. This article reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

References

[1] Lockheed-Martin-Corp., TO 1F–16AM-2-70JG-10-21 – Organizational Maintenance – Engine Removal and Installation – Model F100-PW-220/220E – USAF/EPAF Series – F-16/B Mid-Life Update aircraft, Technical Manual Job Guide. Lockheed Martin Corporation, Bethesda, MD, USA, 2009.

[2] P. Petridis, I. Dunwell, D. Panzoli, S. Arnab, A. Protopsaltis, M. Hendrix, S. Freitas, Game engines selection framework for high-fidelity serious applications, Int. J. Interact. Worlds 2012 (2012) 1–19.

[3] B. Fonseca, H. Paredes, J. Rafael, L. Morgado, P. Martins, Paulo, A software architecture for collaborative training in virtual worlds: F-16 airplane engine maintenance, in: Adriana S. Vivacqua, Carl Gutwin, Marcos R.S. Borges (Eds.), Collaboration and Technology: 17th International Conference, CRIWG 2011, Paraty, Brazil, October 2–7, 2011. Proceedings, 102–109. Berlin, Springer, 2011.

[4] Y.F. Chen, G. Rebolledo-Mendez, F. Liarokapis, S. de Freitas, E. Parker, The use of virtual world platforms for supporting an emergency response training exercise, in: Proceedings of the 13th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games, Wolverhampton, UK, 2008, pp. 47–55.

[5] C. Rodrigues, D. Coelho, L. Morgado, J. Varajão, A. Haidimoschi, G. Doppler, H. Koivusalo, P. Jokinen, G. Velegrakis, C. Sancin, V. Carmenini, Virtual learning for the management of successful SMEs in Europe, in: Leonel Morgado, Nelson Zagalo, Ana Boa-Ventura (Eds.), Proceedings of the SLACTIONS 2009 International Conference – Life, Imagination, and Word using Metaverse Platforms, ISBN 978-972-669-924-8. Vila Real, Portugal: Universidade de Trás-os-Montes e Alto Douro, 2009, pp. 165–170.

[6] M. Boulos, L. Hetherington, S. Wheeler, Second life: an overview of the potential of 3-D virtual worlds in medical and health education, Health Info. Libr. J. 24 (4) (2007) 233–245.

[7] K. Hudson, K. deGast-Kennedy, Canadian border simulation at Loyalist College, J. Virtual Worlds Res. 2 (1) (2009) 3–11.

[8] S. de Freitas, Serious Virtual Worlds. A Scoping Study, JISC, Bristol, UK, 2008.

[9] R. Smith, The long history of gaming in military training, Simul. Gaming 41 (1) (2010) 6–19.

[10] K.A. Orvis, J.C. Moore, J. Belanich, J.S. Murphy, D.B. Horn, Are soldiers gamers? Videogame usage among soldiers and implications for the effective use of serious videogames for military training, Mil. Psychol. 22 (2) (2010) 143–157.

[11] T.M. Sotomayor, Teaching tactical combat casualty care using the TC3 sim game based simulation: a study to measure training effectiveness, Stud. Health Technol. Inf. 154 (2010) 176–179.

[12] P.A. Fishwick, An introduction to OpenSimulator and virtual environment agent-based M&S applications, in: Winter Simulation Conference (WSC '09), 2009, pp. 177–183.

[13] G. Krasner, S. Pope, A cookbook for using the model-view controller user interface paradigm in Smalltalk-80, J. Object Oriented Prog. 1 (3) (1988) 26–49.

[14] C. Stocker, B. Sunshine-Hill, J. Drake, I. Perera, J. Kider, N. Badler, CRAM it! A comparison of virtual, live-action and written training systems for preparing personnel to work in hazardous environments, in: Michitaka Hirose, Benjamin Lok, Aditi Majumder, Dieter Schmalstieg (Eds.), IEEE Virtual Reality 2011 Singapore, March 19–23, 2011, Proceedings, ISBN 978-1-4577-0037-8. Danvers, IEEE, MA, 2011, pp. 95–102.

[15] N. Badler, C. Erignac, Y. Liu, Virtual humans for validating maintenance procedures, Commun. ACM 45 (7) (2002) 56–63.

[16] T. Marsh, Serious games continuum: between games for purpose and experiential environments for purpose, Entertain. Comput. 2 (2) (2011) 61–68.

[17] V. Narayanasamy, K.W. Wong, C.C. Fung, Distinguishing games and simulation games from simulators, ACM Comput. Entertain. 4 (2) (2006) 9.

[18] A. Vilela, A. Marques, H. Costa, J. Rafael, R. Prada, L. Morgado, Aplicação de avatares autónomos para desempenhar o papel de membros na execução de trabalhos em equipa, in: CISTI 2012: Actas de la 7ª Conferencia Ibérica de Sistemas y Tecnologías de Información, 2012, Madrid, Spain, 2012.

[19] R. Smith, B. Hawkins, Lean Maintenance, Elsevier Butterworth-Heineman, Oxford, UK, 2004.

[20] A. Holzinger, M. Kickmeier-Rust, S. Wassertheurer, M. Hessinger, Learning performance with interactive simulations in medical education: lessons learned from results of learning complex physiological models with the HAEMOdynamics SIMulator, Comput. Educ. 52 (2) (2009) 292–301.