# MACHS: an authoring tool to create serious games for machine-tool operator training

A. Mujika[*], D. Oyarzun[*], I. Iparragirre[*], J. Dominguez[**] and J. Arambarri [***]
[*] Vicomtech-IK4, Donostia-San Sebastian, Spain
[**] IDEKO-IK4, Elgoibar, Spain
[***] Virtualware, Basauri, Spain
{amujika,doyarzun,iiparragirre}@vicomtech.org, jdominguez@ideko.es, jarambarri@virtuaware.es

***Abstract*—In this paper, we present the project MACHS, a platform for the generation of serious games, i.e. courses in 3D environments for machine-tool training. The platform consists of two applications: an easy-to-use authoring tool for editing the courses and a 3D simulator for running them. In this paper, we describe the features and the functionalities of each part of the platform. We also describe the structure of the XML files designed for the storage of the information about courses, machines, cameras, interaction, etc. and how we use them to link the editor and the simulator.**

## I. Introduction

Serious Games are generally defined as Computer Games that are not directly oriented to entertainment. The purpose of Serious Games is to use such playful aspect for educational, informative or communicative objectives.

According to experts from the European Center for Children's Products [1], Serious Games can be classified in five categories: Edutainment, Advergaming, Edumarket Games, Political Games and Training and Simulation Games.

While the first is focused on transmitting knowledge, the second uses Serious Games for marketing purposes. Edumarket games are a mixture of the first two types. Political Games' objective is to make aware of political and social topics. Finally, Trainning and Simulation Games immerse the user in an environment whose behaviour replicates a real environment's behaviour, in order the user to familiarize with it. The platform described in this paper was developed to create and run courses that can be classified in the last category.

In recent years, the number of applications in the field of Training Games has increased rapidly. Despite the fact that most of them are in experimental phase, there are some commercial applications that show that Serious Games could be very useful. 3Dsolve [2] and BreakAway [3], that uses Serious Games for military training or PIXELearning [4], that use Serious Games to train adults in business and finance, are some examples.

Most of the work done in the field of Training Games needs experts in two fields. An expert in computer games who develops the application and an expert in the subject to train, who designs the game logic. MACHS, the project presented in this paper, partly funded by Basque Government (IG-2009/0000607), pretends to eliminate the need of the expert in computer games or 3D animation.

The main idea of the project MACHS is that an expert in machine-tool manufacturing could create a course easily and without any help. For that, we developed an authoring tool called Course Editor and a simulator that runs the course designed by the user.

The paper starts with a brief overview of the state of the art in Serious Games in section II. Next, we describe the applications that compose the system MACHS in section III and define the XML files used for their correct running in section IV and V. Finally, we conclude the paper and describe future improvements in section VI and VII.

## II. Related Work

The success of the utilization of Serious Games is clear if we take into account the amount of fields they have been used in.

Health and Medicine are two of the fields where the Serious Games have been used most. Cabas et al. [5] developed an application or training emergency medical services nurses in decision making and Siqueira and Nunes [6] presented a Serious Game to train the users of medical Virtual Reality tools. Göbel et al. [7] introduced some health exergames using a combination of game technology and game-based methods and concepts (e.g. competitive multiplayer features), mechanisms for personalization and adaptation and sensor technology. Besides, it is remarkable that there are several companies [8, 9] dedicated to create Serious Games in this field.

Several projects use Serious Games for training skills in emergency situations. This kind of situation is not easy to reproduce in the real world, but it is in a 3D environment. Chittaro and Ranon [10] propose serious games as a tool to acquire personal fire safety skills and Haferkamp and Krämer [11] presented DREAD-ED, a technology-based teaching methodology for in crisis management units training. Linehan et al. [12] created a serious game designed to teach group decision making skills to coordinators of groups that respond to real-world emergencies such as floods, fires, volcanoes and chemical spills. The results showed that trained groups work better than the non-trained ones. This kind of projects and other different works [13, 14] show that Serious Games are helpful to improve the work in groups.

There are projects that work in really different fields, like management of National Parks for Biodiversity Conservation [15], cultural training [16, 17], language learning [18, 19] or military training [20].

There are some projects that are similar to ours or can be classified in the same field. Games2Train [21] is a shooter type game that aims to train students in the utilization of CAD 3D. The students are inserted in a story where they have to meet several milestones. For example, they have to design some pieces of the machine that will be used to defeat their enemy.

Rilling et al. [22] presented an interdisciplinary work on the application of computer game principles and techniques within an automation industry training scenario.

Rosendo et al. [23] developed a serious game for training of live line maintenance activities, i.e. activities without interrupting the flow of energy on the line. The platform uses devices such as the Nintendo Wii Remote and 3D TV sets to provide a novel model of interaction and navigation.

Regarding authoring tools, the work presented by Göbel et al. [7] contains an application based on [24]. Although it is similar to our Course Editor, the utilization of 3D animation and the machine-tool context make them different.

## III. System Overview

The system developed in the project MACHS contains two main applications. The first application is an authoring tool where an expert in machine-tool manufacturing can design a course for students that are learning how to use a machine or how to do its maintenance. The second application receives the output of the first one and runs a course in a 3D environment where the students must follow the steps designed by the expert. In what follows, we will denote these two applications Course Editor and 3D Simulator respectively. Furthermore, the students have two options in the 3D Simulator: they can run the simulation, where the system shows the correct steps automatically or they can do an exercise where they have to explore the 3D reproduction of a machine and try to take the correct steps to finish the course correctly. The system supervises the exercise and displays texts about the correctness of the user's performance.

### *A. Course Editor*

As stated before, in this part, the teacher, the person that knows the machine and how it works, prepares a course about the control or the maintenance of the machine. The main goal of the Course Editor is its usability, so that an expert in machine-tool manufacturing but not necessarily initiated in 3D applications can generate a 3D course easily.

The user of this application can edit some general features that describe the course:

- Name: the name of the course will not appear in the 3D Simulator, but it identifies the course.
- Description of the simulation: the user defines the statement that will be showed during the simulation, i.e. the option that shows the correct steps.
- Description of the exercise: the user defines the statement that will be showed during the exercise.

The course is designed in the main diagram of the Course Editor (see figure 1). The diagram is composed by squares that are linked with arrows. Each item or square in the diagram represents a step that can be taken during the course and the arrows represent if it is allowed to concatenate these steps. The user of the editor has to define the name of the step, which piece has to be chosen and which action has to be done with it, e.g. the user defines the step "tighten screw 1" by choosing the "screw 1" in the module "carriage X" and the action "tighten". Moreover, the teacher can set the statement that will appear when this step is taken, e.g. "You tightened screw 1. This is a correct step".

All the items show the name of the action in order the teacher to catch the structure of the diagram, i.e. the course, in a quick look.

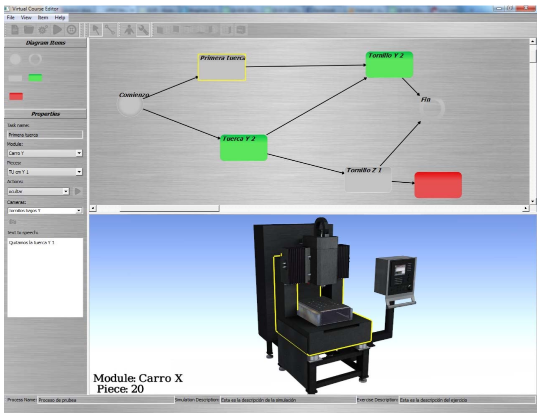


Figure 1. The diagram and the machine in the Course Editor

These are the different types of items that can be defined in the Course Editor:

- Initial and final steps: the user can define as many initial and final items as he wants in the diagram. These items are identical with the usual step items. However, the system will identify these steps and will not allow the student to start a course with a step not defined as an initial one and will notify the student that the course is satisfactorily finished when a final step is reached.
- Simulation steps: the teacher can define several paths that lead to a correct end in the diagram, but the 3D Simulator can show only one of those when the student asks the system to do so. Therefore, the teacher defines the path of steps that the 3D Simulator will show with these items. The editor ensures that the chosen path is unique. Simulation items are similar to the usual ones, but in this case, besides the usual steps' aspects, the user has to define the camera used to visualize this action. During the simulation in the 3D Simulator the system handles the camera automatically, so the camera for each step has to be defined in advance. Moreover, the user can isolate the simulation path, i.e. he can remove temporarily the items that are not in the simulation path in order to avoid confusion when designing the simulation.
- Incorrect steps: when the student is doing the exercise, if he doesn't choose an action that follows the last correct step he took, the system automatically tells him he is choosing an incorrect action. Nevertheless, the teacher can set special incorrect steps in the diagram. In these items the statement that will appear in the 3D Simulator is

very important, e.g. "You have just broken the machine".

In order to get a better usability, the editor shows a 3D reproduction of the machine that will be used in the 3D Simulator (see figure 1). The user can move the camera to see the pieces of the machine from any point of view or can use the predefined ones, i.e. the ones that can be chosen when running the 3D Simulator. This aspect is important, because the teacher has to decide which camera will be used during the simulation in the 3D Simulator.

The 3D reproduction of the machine can also be used for choosing the pieces. Instead of choosing the module and the piece in a menu, the teacher clicks the piece in the machine and the system automatically sets the details in the menu. The teacher also can visualize the animation that will be triggered in the 3D Simulator when this action is chosen.

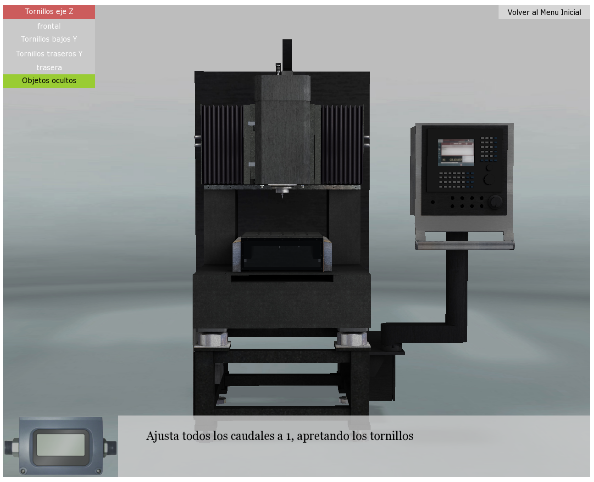
In conclusion, with the overall view of the course showed by the diagram and the help of the 3D reproduction of the machine, the teacher can easily design the course for his students.

### *B. 3D Simulator*

The 3D Simulator is the application that takes the output of the Course Editor and automatically generates the designed course in a 3D environment. The user of the 3D Simulator cannot change anything in it. He can only interact with the application.

Once the user, a student, opens the application, a menu shows two options. The user can ask the system to show the correct steps he has to take to finish the exercise or he can directly try to do the exercise. At any time, the user can return to the main menu and change the mode.

In the first case of the main menu, the system shows all the simulation steps set by the teacher in the Course Editor. The system synchronizes the triggered actions and the points of view changes as defined in the Course Editor, in order the user to watch the steps properly.



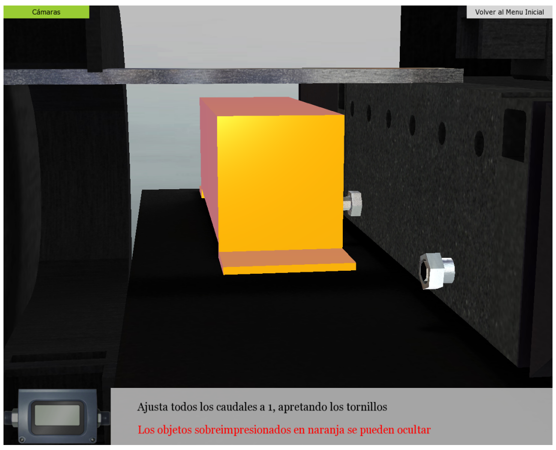
Ajusta todos los caudales a 1, apretando los tornillos

Figure 2. The menus for points of view and hidden objects

In the second case, the user must interact with the machine without any help. For that, he can explore the 3D environment, changing the point of view of the camera. A small menu (see figure 2) in the upper left side of the screen shows all the options. These points of view are predefined and stored in an XML file that describes the machine as we will see later. Paths between different points of view are also predefined. This way, the camera doesn't jump directly to the new point of view and the student doesn't get lost when changing the camera.

Once the user chooses the exercise option, he has to click in the pieces in the correct order. Each click (left or right) represents an action, e.g. left-clicking a screw represents the action of tightening it and right-clicking represents the action of loosening it. During the exercise, if the user chooses the correct step designed in the diagram of the Course Editor, the system displays the texts set for each action and triggers the animation linked to it. If he doesn't choose the right step, i.e. an action that doesn't follow the current one in the diagram of the Course Editor, the system displays a predefined text that tells the user he is making a mistake. As seen before, the selected action can be an incorrect one. In this case, the text defined in the Course Editor is displayed.

To improve the interactivity with the machine, the pieces that the user can interact with are highlighted when the cursor is on them (see figure 3). The user can also remove some pieces temporarily in order to watch other pieces correctly. Removed pieces appear in a small menu (see figure 2) and can be returned to their places. Moreover, removing a piece can be a part of the course.



Ajusta todos los caudales a 1, apretando los tornillos

Los objetos sobreimpresionados en naranja se pueden ocultar

Figure 3. A highlighted piece in the 3D Simulator

In conclusion, the student can watch the steps of the designed course and repeat them easily with the help of the system.

## IV. EDITOR/SIMULATOR CONNECTION

As the teacher, the user of the Course Editor is not supposed to be an expert in 3D animation. He will not be able to change any aspect of the 3D simulator, only the file generated by the Course Editor. Thus, the Course Editor and the 3D Simulator have to be perfectly synchronized in order the 3D Simulator to reproduce exactly what has been designed in the Course Editor.

In order to achieve this synchronization, we define an XML file denoted Virtual Course Simulation (VCS). This file is created by the Course Editor exactly in the way the 3D Simulator will be able to obtain the information. Figure 4 shows the structure a VCS file.

```
<vcs>
  <exercise>
    <action>
       <transition/>
       <transition/>
       …
    </action>
    …
  </exercise>
  <simulation>
    <action>
       <transition/>
    </action>
    …
  </simulation>
</vcs>
```

Figure 4. Overview of a VCS file

The root of the document tree contains the name of the course and the path of the file to open the corresponding machine.

```
<vcs name="example" machine="myMachine.xml">
```

There are two main elements in the document: *exercise* and *simulation*. The element *exercise* contains exercise's description in its attributes:

```
<exercise description="Balance all the flows">
```

The element *exercise* contains elements that contain the information about the steps that must be taken during the exercise and called *action*.

```
<action id="9" name="tighten screw"
type="action" module="carriage Y" piece="screw"
action="tighten" speech="This is a correct
step">
      <transition id="12" />
      <transition id="49" />
</action>
```

The attributes of the element are *id*, *name*, *type*, *module*, *piece*, *action* and *speech*. The attribute *type* can be *start*, *action*, *end* or *incorrect*, as described before. The attribute *speech* is the text that will be displayed when taking this step. The elements called *transition* show which step can be taken after the current one. Any action that doesn't follow the current one makes the system to display a predefined error text. However, an allowed action can be an incorrect one and the system will display the error text, defined in the attribute *speech*.

The element *simulation* comes after the element *exercise* and it is very similar.

```
<simulation description="the simulation">
```

In this case the elements that contain the information about the steps have a new attribute. The attribute *camera* indicates the id of the point of view used to watch the current action.

```
<action id="0" name="tighten screw 1"
type="start" module="carriage Y" piece="screw 1"
action="tighten" speech="now we tighten the
screw" camera="4">
      <transition id="2" />
</action>
```

Note that the actions in the element *simulation* can only have one transition, since simulation is a sequence of actions with no different options.

In summary, the VCS file contains all the information about the diagram designed in the Course Editor classified tidily. The element *action* contains all the information concerning the squares of the diagrams and the element *transition* all the information concerning the arrows.

## V. MACHINE DESCRIPTION

The VCS file is essential for synchronizing the Course Editor and the 3D Simulator, but it is not less important that both applications work with the same machine. Besides using the same 3D model, it is necessary that the machines used in both applications have exactly the same characteristics, i.e. all the modules, pieces and actions defined in the Course Editor must be defined in the 3D Simulator.

Therefore, we defined another XML file, called Virtual Machine Markup Language (VMML), which contains all the information needed to understand the machine's functionality (see Figure 5).

```
<machine>
  <module>
    <piece>
       <action>
          <callback/>
          <translate/>
          <rotation/>
          …
          <hide/>
          <measure/>
          …
       <action/>
       …
    </piece>
    …
  </module>
  …
</machine>
```

Figure 5. Overview of a VMML file

The root of the file contains the path of other files that contain information about the predefined points of view and measures related to the machine and its interaction.

```
<machine name="myMach" cameras="MyCameras.xml"
measures="measureHUD.xml">
```

In this case, the file located in the path defined in the attribute *cameras* contains the predefined points of view for the camera and the paths that the camera will follow when changing the point of view.. The file described in the attribute *measures* defines the Head-Up Display

(HUD) that shows measures linked to some pieces. It contains HUD's placement, aspect, piece names and initial values. If new similar files were needed, the element machine would contain their paths.

The main elements of the XML file are *modules* and *pieces*. The structure of the file follows the structure of the machine, i.e. if a module contains a piece in the machine the element that corresponds to the module contains the element that corresponds to the piece. This structure is important in order the system to animate all the pieces or the modules that are inside a module when animating such module. For example, it is usual that a machine has a carriage in each axis; X, Y and Z. If carriage Y's movement is dependent of carriage X's movement the structure of the file will be:

```
<module name="carriage X" file="carX.3DS">
    <module name="carriage Y" file="carY.3DS">
        <piece name="screw 1">
```

Note that the attribute *file* can appear in any module and piece. The geometry of a module or a piece can be saved in a specific file or it can be part of a file defined in a higher level. In the second case, it is crucial that the name of the piece inside the 3D model is the same as the one defined in the VCS. Furthermore,

Each *piece* element contains the list of actions that can be done interacting with it.

```
<action    name="hide"    type="click_right"
duration="1.0">
```

where the attribute *type* indicates the interaction type, e.g. click_left, 'a', etc. and the attribute *duration* sets the duration of the action. This number is not needed during the exercise, but when running the simulation, it indicates how long the system has to wait until it triggers the next action. It is important that the duration is well defined. If the duration is too small, a premature change of point of view can make the action invisible and if the duration is too high, the simulation can be boring for the user.

Inside the element *action*, the file defines what happens when activating such action.

- The element *callback* links an animation saved in a file to the action. Note that it is not necessary that the animation works in the current piece, e.g. in the following case the animation works in the module "carriage Y". This characteristic could be applied to any of the following elements.

```
<callback  module="carriage  Y"  piece="NULL"
file="carriageYcallback.txt" />
```

- The element *translate* defines the translation that will be applied to the piece when activating the action.

```
<translate x="0.0" y="0.0" z="0.1" />
```

- The elements called *rotation* define the rotation that will be applied to the piece by giving an angle and an axis. In this case, it could be useful that the system accepts more than one element of this type.

```
<rotation axisX="0.0" axisY="0.0" axisZ="1.0"
angle="-0.5"/>
```

- The element *hide* sets that this piece will be removed from the 3D machine when activating the current action. Moreover, it is possible to define the animations that will be triggered when hiding the piece or when it returns to the 3D scene. These animations can be very helpful for obtaining a more impressive result. For example, when removing screw, an animation that represents its rotation can be triggered, making the action more realistic and more helpful for the student.

```
<hide          hidecallback="hideScrew.txt"
returncallback=" returnScrew.txt"/>
```

- As stated before, in some machines, moving some pieces makes some parameters to change. The VMML file takes this aspect into account and defines an element called *measure*. The change applied to the parameter is defined by the attributes. Sums, products, exponentials and even value tables have been implemented, but the system is prepared to easily add a new formula.

```
<measure name="caudal Z1" product="0.95" />
```

The main goal of the VMML file was the simplicity of its structure so as to easily add new features required by a new machine. So, besides the possibility of designing any structure of modules and pieces, without losing any information about the structure of the machine, any new feature can be added inside the element *piece* and any new action can be added inside the element *action*.

## VI. Conclusion

In the project MACHS we developed a system that makes creating Serious Games easier. The user of the Course Editor can design the course without taking into account any aspect related to 3D Animation or Computer Science.

The user designs the diagram that represents the course that the students will run in the 3D Simulator and doesn't need to work in the 3D Simulator. It reproduces the designed course automatically.

Several aspects of the Course Editor make it a friendly interface. Mainly, the 3D reproduction of the machine helps the user creating the desired diagram. The 3D scene inserted in the authoring tool highlights the chosen piece and shows the selected point of view in order the user to have the possibility to check he has chosen the correct features for the course. The 3D scene also reproduces the chosen actions.

The election of the pieces can be made by choosing the name in a menu, but it can also be made by clicking in the 3D machine.

For the synchronization of the Course Editor and the 3D Simulator we designed two new XML files called VCS and VMML. The VCS file contains the information saved in the diagram of the Course Editor and the VMML file describes the structure of the machine, its components and their behaviour. Despite the fact that both files have a simple structure, they contain all the information needed for the correct running of the system.

Moreover, the simplicity of the structure of the files makes them valid for future improvements in the system. The structure of the files is designed to add new elements easily and without losing the information already inserted.

## VII. Future Work

The project MACHS is only the first step of a long way that we intend to walk. Therefore, there are several aspects that need to be improved in the system and there are some features that we want to incorporate in the system.

Regarding the Course Editor, instead of working only with items, we plan to implement group of items. We reckon that working with subdiagrams inside the main diagrams will help the user understand the structure of the course. Besides, the user will be able to watch the designed simulation inside the Course Editor, not only the actions separately. This way, the user will be able to evaluate his work without opening the 3D simulator.

Regarding the 3D Simulator, our experience in 3D animation leads us to believe that using virtual characters is really helpful for this kind of applications. The interface becomes friendlier if there is an avatar talking with the student instead of a HUD that shows the text. Moreover, during the simulation, the virtual character could show how to do each action, e.g. the avatar shows how to operate a part of the machine and which tool to use. This way, students' learning would be much better.

The incorporation of a virtual character in the system leads to a higher complexity of the XML files designed for the project MACHS. Nevertheless, we believe that the simple structure of such files makes easy to add new features.

Finally, we see that the system developed in the project could have another application that make it complete. Now, we assume that the user of the Course editor, the teacher, doesn't have to do anything with the definitions of the modules, the pieces and their interaction options, i.e. the VMML file. However, we plan to develop an application called Machine Editor that will allow the teacher to define the VMML file easily.

This way, the teacher could take any 3D geometry and define its behaviour. Then, in the Course Editor he could have exactly the desired machine, not the one designed for others. So, the designed courses would be perfectly adapted to students' needs.

## References

[1] J. Alvarez, and O. Rampnoux, "Serious Game: Just a question of posture?", *Artificial & Ambient Intelligence, AISB'07*, pp. 420–423, April 2007

[2] 3Dsolve Home Page, *www.3dsolve.com*

[3] BreakAway Home Page, *www.breakawaygames.com*

[4] PIXELearning, *www.pixelearning.com*

[5] A. Cabas Vidani, L. Chittaro, and E. Carchietti, "Assessing Nurses' Acceptance of a Serious Game for Emergency Medical Services", *VS-GAMES '10 Proceedings of the 2010 Second International Conference on Games and Virtual Worlds for Serious Applications,* 2010

[6] R. Siqueira, and F. L. S. Nunes, "Applying Entertaining Aspects of Serious Game in Medical Training: Systematic Review and Implementation", *SVR '11 Proceedings of the 2011 XIII Symposium on Virtual Reality*, 2011

[7] S. Göbel, S. Hardy, V. Wendel, F. Mehm, and R. Steinmetz, "Serious games for health: personalized exergames", *Proceedings of the international conference on Multimedia*, 2010

[8] ElderGames Home Page, *www.eldergames.org/*

[9] MIMICS Home Page, *www.mimics.ethz.ch/*

[10] L. Chittaro, and R. Ranon, "Serious Games for Training Occupants of a Building in Personal Fire Safety Skills", *VS-GAMES '09 Proceedings of the 2009 Conference in Games and Virtual Worlds for Serious Applications*, 2009

[11] N. Haferkamp, and N. C. Krämer, "Disaster readiness through education - training soft skills to crisis units by means of serious games in virtual environments", *EC-TEL'10 Proceedings of the 5th European conference on Technology enhanced learning conference on Sustaining TEL: from innovation to learning and practice*, 2010

[12] C. Linehan, S. Lawson, M. Doughty, and B. Kirman, "Developing a serious game to evaluate and train group decision making skills", *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, 2009

[13] A. Hogue, B. Kapralos, and T. Pierce, "A serious game for collaborative intercultural business communication", *Proceedings of the 13th International Conference on Humans and Computers*, 2010

[14] M. Q. Tran, and R. Biddle, "Collaboration in serious game development: a case study", *Future Play '08 Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 2008

[15] E. Vasconcelos, C. Lucena, G. Melo, M. Irving J.P. Briot, V. Sebba et al., "A Serious Game for Exploring and Training in Participatory Management of National Parks for Biodiversity Conservation: Design and Experience", *SBGAMES '09 Proceedings of the 2009 VIII Brazilian Symposium on Games and Digital Entertainment*, 2009

[16] M. A. Zielke, M. J. Evans, F. Dufour, T. V. Christopher, J. K. Donahue, Phillip Johnson et al., "Serious Games for Immersive Cultural Training: Creating a Living World", *IEEE Computer Graphics and Applications, Volume 29 Issue 2*, March 2009

[17] M. Hays, H. C. Lane, D. Auerbach M. G. Core, D. Gomboc and M. Rosenberg, "Feedback Specificity and the Learning of Intercultural Communication Skills", *Proceeding of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling*, 2009

[18] W. L. Johnson, "Serious Use of a Serious Game for Language Learning", *International Journal of Artificial Intelligence in Education archive Volume 20 Issue 2*, April 2010

[19] W. L. Johnson, H. Vilhjalmsson, and S. Marsella, "Serious Games for Language Learning: How Much Game, How Much AI?", *Proceeding of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, 2005

[20] A. Sanchez, and P. A. Smith, "Emerging technologies for military game-based training", *SpringSim '07 Proceedings of the 2007 spring simulation multiconference - Volume 3*, 2007

[21] Games2train Home Page, *www.games2train.com*

[22] S. Rilling, U. Wechselberger, and S. Mueller, "Bridging the Gap Between Didactical Requirements and Technological Challenges in Serious Game Design", *CW '10 Proceedings of the 2010 International Conference on Cyberworlds*, 2010

[23] M. Rosendo, T. Buriol, K. de Geus, S. Scheer, and C. Felsky, "Towards the Development of a 3D Serious Game for Training in Power Network Maintenance", *VS-GAMES '11 Proceedings of the 2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, 2011

[24] S. Göbel, L. Salvatore, R. Konrad, and F. Mehm, "StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories", Proceedings of the 1st Joint Int'l Conference on Interactive Digital Storytelling, ICIDS pp. 325–328. 2008