Visual Comput (2008) 24: 719–726
DOI 10.1007/s00371-008-0252-x

ORIGINAL ARTICLE

Alejandra García-Rojas
Mario Gutiérrez
Daniel Thalmann

Visual creation of inhabited 3D environments

An ontology-based approach

Published online: 17 May 2008

A. García-Rojas (✉) · D. Thalmann
Virtual Reality Laboratory,
EPFL, Switzerland
{alejandra.garciarojas,
daniel.thalmann}@epfl.ch

M. Gutiérrez
INAOE,
Puebla, México.
mgutierrez@ccc.inaoep.mx

Abstract The creation of virtual reality applications and 3D environments is a complex task that requires good programming skills and expertise in computer graphics and many other disciplines. The complexity increases when we want to include complex entities such as virtual characters and animate them. In this paper we present a system that assists in the tasks of setting up a 3D scene and configuring several parameters affecting the behavior of virtual entities like objects and autonomous virtual humans. Our application is based on a visual programming paradigm, supported by a semantic representation, an ontology for virtual environments. The ontology allows us to store and organize the components of a 3D scene, together with the knowledge associated with them. It is also used to expose functionalities in the given 3D engine. Based on a formal representation of its components, the proposed architecture provides a scalable VR system.
Using this system, non-experts can set up interactive scenarios with minimum effort; no programming skills or advanced knowledge is required.

Keywords Inhabited virtual environments · Visual programming · Authoring tool · Ontologies

1 Introduction

Computer games such as those from EA games [5] offer users the possibility to create and customize inhabited scenarios in their worlds. They provide GUI-based authoring tools that are easy to use. However, these environments have limited features, and only expert developers are able to take advantage of the underlying game engine. Virtual environments (VE) have many other applications besides entertainment, for example, the preservation of cultural heritage, simulation of crowd behavior, simulation of architectural buildings, training, etc.

Creating these applications requires experience in computer graphics, human-computer interaction, artificial intelligence, animation techniques, etc. In most cases good programming skills are also needed to put together all these components. The fact that one relies on expert programmers when implementing a virtual reality (VR) application can be an obstacle for designers and other creative members of the production team [2].

Various software libraries and development environments have been created to ease the integration task mentioned above. However, in the best of the cases, programming skills are still required in order to script animation sequences, set up interaction paradigms or program autonomous characters.

In this paper we propose the use of a visual programming paradigm that allows us to set up an interactive vir-

tual environment with autonomous characters without the need of writing a single line of code. Other systems use similar visualization techniques, but in most cases they are limited to providing visual aids to understanding low-level relations between entities such as hierarchies (skeletons), and other properties (textures, materials).

Our visual programming language aims to represent different components of a 3D scene as a 2D abstraction, reducing the complexity of relationships between entities and concepts inside a VE.

Our approach is based on a semantic representation of the components of a virtual environment: virtual entities and animation algorithms, the latter including behavior controllers for autonomous virtual humans. Our visual programming interface is supported by an ontology for virtual environments and allows us to represent spatial relations between objects in the scene, and connect animation controllers to virtual objects.

The paper is organized as follows: the next section describes some authoring tools for virtual environments; we discuss how they represent the components of a 3D scene and what kind of information/knowledge they present to the user. Section 3 describes the semantic representation supporting our system; Sect. 4 describes the representation of virtual entities using visual programming. In Sect. 5 we present the application, and finally our conclusions and future work.

2 Related work

Some domain-specific tools have been created to assist non-experts in the creation of virtual worlds. As an example, Costagliola et al. [4] proposed an authoring tool for creating virtual exhibitions. This tool uses text-based or iconic-based interfaces to set up scenarios with basic 3D objects.

Generic tools propose the use of scripting and XML languages [10, 13]. Green et al. [6], presented an authoring tool that accelerated the creation of content for 3D environments using Python scripting with some extension mechanisms. The development framework presented in [15], also uses Python scripts to configure different libraries and extend the system functionality. The developer is able to create scenes populated with autonomous virtual humans by means of short programs (Python scripts). A problem with scripts arises when using wrap functions, like those implemented in Python: we need a good documentation of functions, and users have to become familiar with the initialization and internal workflow of the 3D engine. This represents an important learning curve, once again, people interested in creating a VR application need some programming skills.

Commercial systems such as Virtools [17] and Quest 3D [1] offer a graphical schematic interface to navigate and manipulate entities in the 3D scene. This is in fact a visual representation of the scene graph: a hierarchical data structure representing the spatial relationships between 3D objects.

Three-dimensional modeling and animation tools such as Maya and 3DS Max, also provide this kind of scene graph visualization, which are helpful when managing the components of a complex scene.

As we have mentioned, these kinds of visual representations focus on the geometric aspects of virtual entities. These representations allow us to see how elements like virtual characters are composed (skeleton, hierarchy of joints and segments, etc.) and change some of their properties (size, position, texture, etc). Complex information, such as animation algorithms, can be manipulated only through scripts. In order to set up interactive behavior the author still needs a fair knowledge of computer programming.

To overcome the necessity of specific knowledge, we propose an intermediary representation that captures relevant concepts and knowledge associated with the creation of a virtual environment. Ontologies have been successfully applied to represent the knowledge and concepts of specific domains. The description of 3D items using an ontology has already been described in several works. For instance, the X3D ontology [9] represents very specific and low-level concepts associated with a 3D scene.

Ontologies also have the power to represent high-level properties and knowledge associated with the entities composing a VE, including geometric properties. For example, recent applications in product design aim at capturing conceptual functionalities associated with 3D shapes, supporting collaboration in the design process [3].

Considering high-level properties is very important when developing a virtual environment. When we define the behavior of virtual humans, we are not interested in the number of bones and degrees of freedom (DOF) of a character, we rather care about its personality, the description of its emotional state and the animations that can be used to express/reflect these properties.

The next section describes the approach we have followed to represent high-level properties and knowledge associated with entities in a VE. When supported by a visual programming interface, this approach can help VE designers to create more complex environments with less effort.

3 Semantics for inhabited 3D environments

We consider that an authoring tool for creating virtual environments should be flexible, scalable and adaptable to different needs. Gutiérrez et al. [7] developed the concept of semantic virtual environments with the goal of creating environments that can reuse digital items and be scaled in functionality.

This semantic representation is very helpful in describing complex environments based on the semantic description of their 3D components. An example of a high-level description of complex entities is the virtual human ontology [8]. This ontology aims at describing virtual humans as an active semantic entity with features, functionalities and interaction skills. For this work we have partially used this ontology.

The development of an ontology that describes the 3D scene is a crucial task of this work. The information that an ontology can provide is more than just data, it describes concepts and their properties, and relations with other concepts. Our application uses this information to expose higher-level features to the non-expert user.

The ontology we created is presented in Fig. 1. It was developed in Ontology Web Language (OWL) [18] using the Protégé software [16]. The main concept, or class, is *resource*; it represents the items that the user can place in the 3D scene. We have three kinds of resources: virtual humans, objects and scenarios. Virtual humans have three main properties: *hasIndividualDescription* (emotional state and personality), *hasMorphologyDescription* and *canPerformAction*. *VirtualHumanActions* are human like capabilities, and they are related to a *Controller* which is a description of the implementation of the action (e.g., library) available in the 3D engine. *VirtualHumanActions* can be related to other resources, for example *VirtualHuman hasAction LookAt targets VirtualHuman*.

Figure 2 presents an example of the virtual human action *DanceAction*. This action has the property *usesAnimation* with the restriction *some Animation that hasCategory Dance*. The controller that this action uses is *KeyFramePlayer* whose function has the parameters *VirtualHumanName* and *AnimationFile*.

The parameters of a controller are also described, and have the following properties that are dependent of the action they are used for:

propertyName gives the name of the action property that establishes the relationship between the instance and the property of interest.
classNameSource is the name of the class from where the parameter will be taken.
propertyNameSource is the name of the property that has the value for the parameter.

The parameter descriptions are used to send messages from the 2D to the 3D engine. This is described in the following section.

The ontology also describes other constrains that can be used as filters in the application. For example, if a virtual human has a defined emotional state, then the animations to

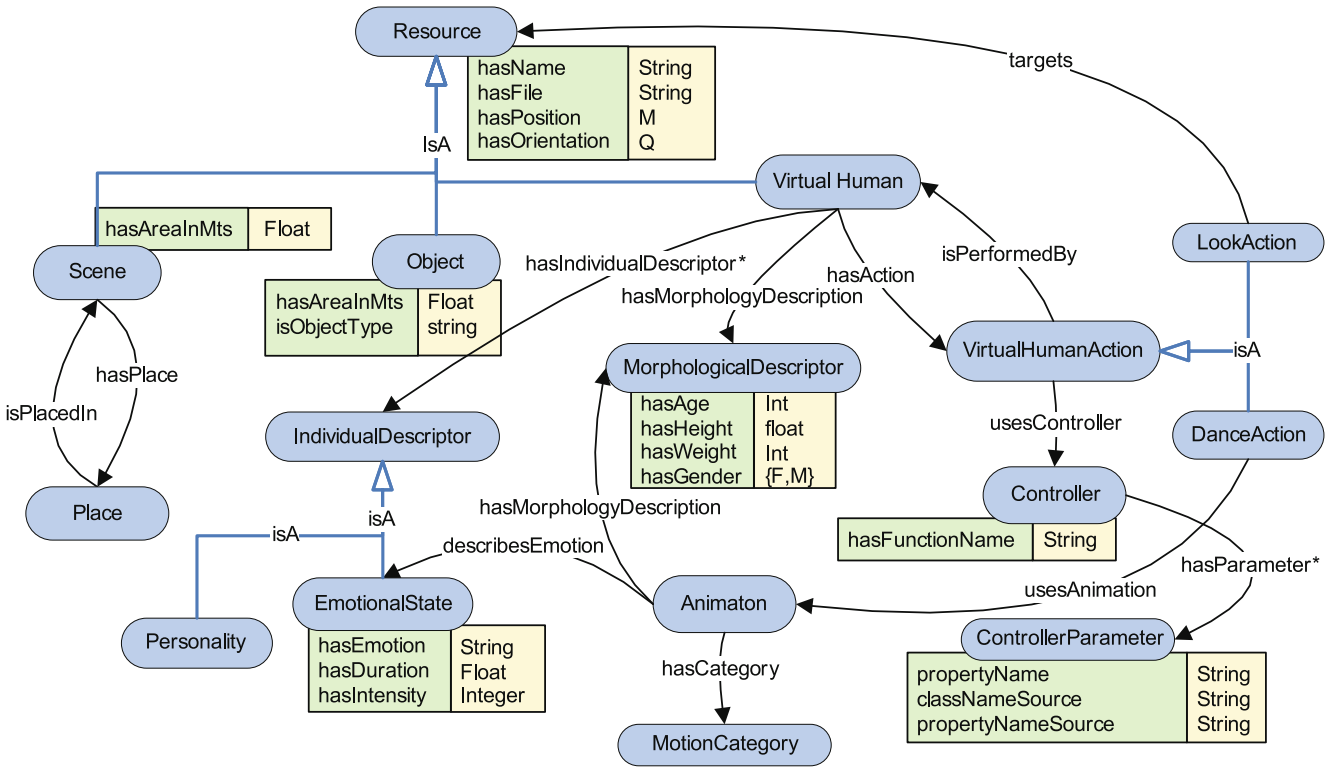

Fig. 1. Ontology diagram

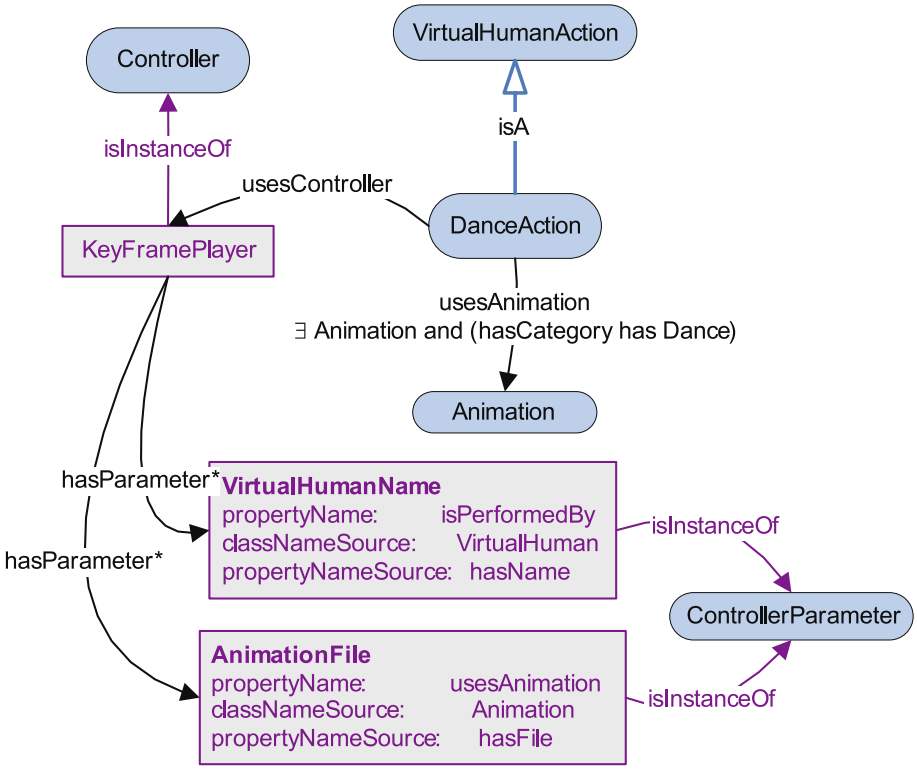

Fig. 2. Description of virtual human actions and their controllers in the ontology

choose are limited to those that express the same emotion. Something similar happens with the morphology description, the animations available for a specific character are limited to those suitable for its particular morphology.

Using the ontology we are able to define all functionalities provided by the 3D engine under use. Different 3D engines can be utilized, all we have to do is add to the ontology the instances corresponding to the specific controllers. In the following section we describe how these functionalities are exposed and used in the visual programming language.

4 System description

This section describes the visual programming language used to represent the 3D scene with the help of the ontology. Followed by the system architecture.

4.1 Visual programming language

Visual programming languages allow us to manipulate elements graphically instead of using text. Elements commonly used include: boxes, arrows, cycles, etc. The implementation we propose is not hierarchical as is the case of the applications described in Sect. 2. Our visual representation consists of a 2D projection of the 3D scene and includes a representation of the interactions between elements, which is closer to the ontology representation.

Figure 3 shows a screenshot of the interface created using the NetBeans Visual Library [12], which offers support for visual modeling. In this scene the main component is the GraphScene, on top of which we add widget layers. Each of those layers has a different purpose: adding widgets, making connections, defining the background, moving widgets, etc.

When the ontology is loaded we get the instances of the subclasses derived from the *resource* class. We display

Table 1. Communication protocol between visual and 3D applications

Function	Message
Quit	0
Add resource	1:name:source: orientation:posx:posz
Remove resource	2:name
Modify resource properties	3p:name:posx:posz 3y:name:posy 3r:(x/y/z):name:angle
Add/remove action	4(a/r):functionName:parameters
Play animation	5

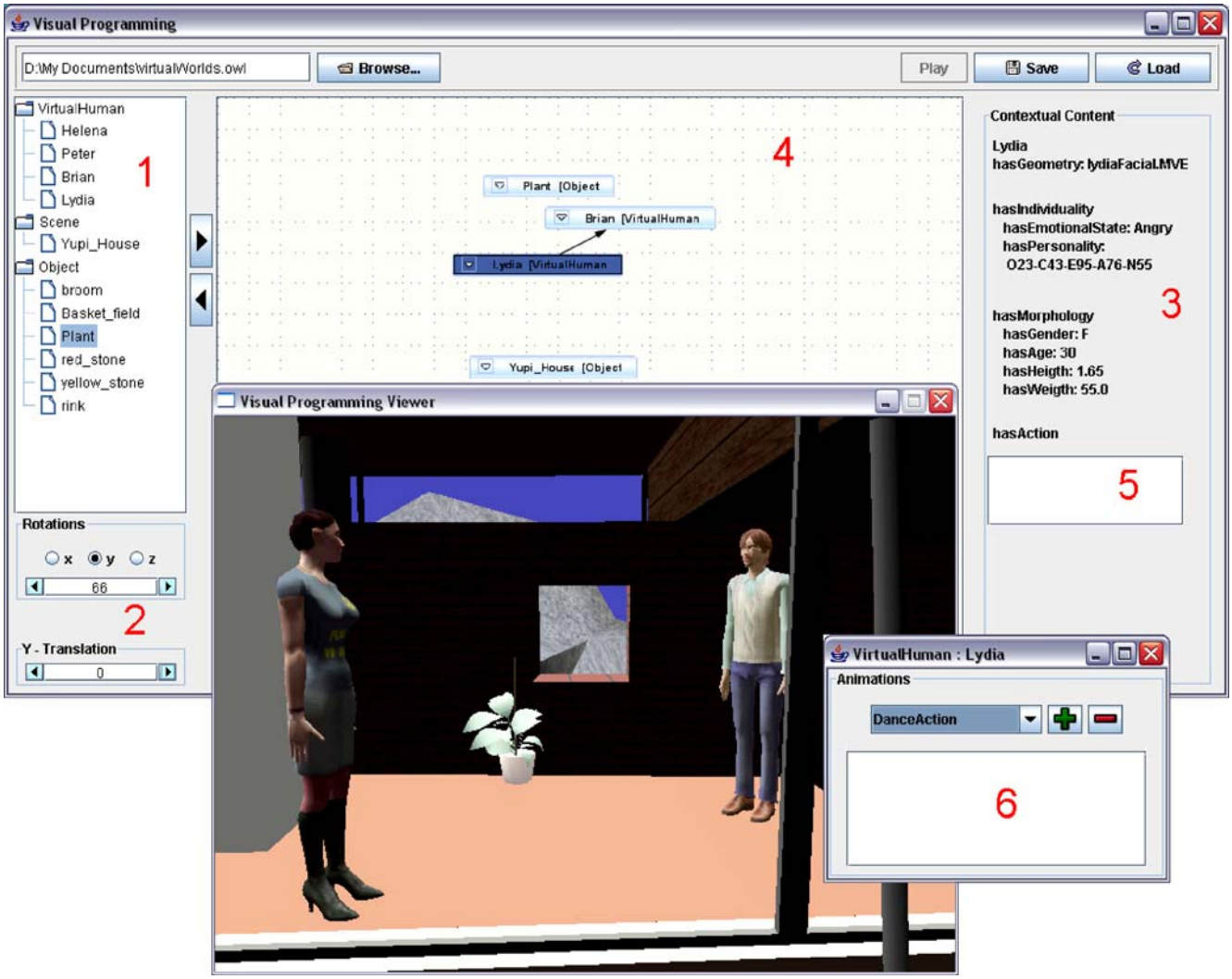

Fig. 3. Two-dimensional representation using the visual programming interface. (1) Available resources. (2) Controllers for y position and rotations. (3) Properties of selected resource. (4) Connection layer to define actions between resources using arrows. (5) List of actions. (6) Independent actions

the available resources as a tree corresponding to the class hierarchy (1 in Fig. 3). The user can select a resource from the tree and place it into the scene. This action produces the creation of a widget in the 2D scene, and it is loaded and rendered in the 3D scene in real time. The bidimensional space of widgets represents the $[x, z]$ plane of the 3D scene; thus, the user can place objects intuitively as if looking at the scene from above. For controlling the rest of the transformations (y position and rotations), the user can use the provided controls (2 in Fig. 3).

Each subclass derived from the resource class also has specialized properties that describe them. When a resource is selected, its properties are displayed in the right box of the interface (3 in Fig. 3). First, we display data properties with their values, and after we display recursively object properties and their values.

To define actions between a virtual human and other resources, the user can draw arrows between them (4 in Fig. 3) through the connection layer. Actions that are not related with other resources are displayed when the user double clicks on them (6 in Fig. 3). Each time the user creates an action for a virtual human, the action is stored in a queue. When the user plays the scene, the scheduled actions are performed (5 in Fig. 3).

Each time the user interacts through the visual programming interface, the system sends messages to the 3D engine in real time. Thus, we have defined a communication protocol presented in Table 1. Messages for the programmed actions for the virtual humans are created using the controller's description from the ontology. For example, if the user creates a relationship *VirtualHuman hasAction Interact Gaze with Object Plant*, the message is

the concatenation of the function's parameters described in the ontology.

The last functionality provided by the system is saving and reloading the scene. This is performed serializing the current items created in the scene into a file that can be subsequently loaded.

In the following we describe the technical details of our implementation.

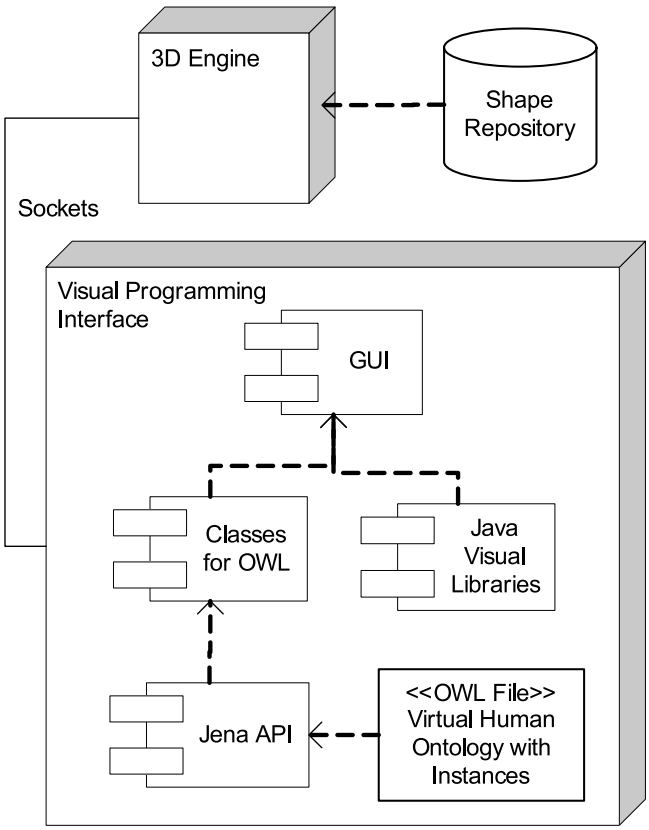

Fig. 4. System architecture

4.2 System architecture

To access the ontology we have used the Jena API [11]. In the visual programming interface we have created classes to represent concepts from the ontology (OWL Classes). These classes are rendered in the GUI using the open source NetBeans Visual Library [12], and can be manipulated with the visual programming language as described before. The 3D engine that we are using [14] supports loading virtual humans and has several functionalities such as key frame animation player and gaze controller. In Fig. 4 we present a diagram of the system architecture.

The modularity of our system makes the 2D interface completely independent from the 3D engine; we just need to implement the messages passed by the interface application and describe actions and controllers in the ontology. However, the 3D engine should support the functionalities previously defined in the ontology.

5 Creating a 3D scene

The creation of a 3D scene requires an ontology populated with the instances of 3D entities, such as virtual humans and 3D objects available in a data repository. The ontology should also contain instances describing the controllers (functions) provided by the 3D engine to be used.

In our 3D engine we have two controllers available: *KeyFramePlayer* and *GazeController*. We have defined the following actions: TalkAction, IdleAction and LookAtAction. These actions can be performed using the defined controllers with the restrictions described in Sect. 3. Talk and idle actions use the *KeyFramePlayer* controller, and use animations belonging to the talk and idle categories, respectively. The animation to play is taken randomly from the available animations that conform to the defined restrictions.

Once the ontology has been set up with the information, it can be loaded in the application. Now, we can

Fig. 5. Animation sequence of virtual humans. *Left* image shows that Lydia talks and Peter is idle. *Center* image shows that Peter looks at the plant. *Right* image shows that Lydia looks at the plant

start placing objects and virtual humans, and programming actions. In our example application we used the following resources: two virtual humans, Lydia and Brian, and two 3D objects, YupiHouse and Plant. We positioned the resources in the desired place inside the house. We programmed actions for Lydia: TalkAction and LookAtAction Plant; and for Brian: IdleAction and LookAtAction Plant. When we play the animation we get the sequence shown in Fig. 5.

Setting up this scene does not take more than a few minutes. Until now, the application allows for creating quite simple scenarios. However, the user can define many different actions depending on the controls they have implemented in the 3D engine and exposed through the ontology.

There are still some issues that can be solved in different ways, e.g., handling animation duration, or animation blending, setting up a time-line, or having an event-based animation system, etc.

6 Conclusions and future work

In this paper we have presented a preliminary version of an authoring tool for virtual environments, based on a visual programming paradigm supported by an ontology for virtual humans and 3D objects.

The system was designed to facilitate the creation of inhabited 3D environments, in particular for users with no or minimal programming skills.

Our tool is supported by an ontology that can describe 3D scenes inhabited by virtual humans with various features, such as animations dependent on the personality and emotional state of the virtual character.

The visual programming paradigm is used to represent 3D entities and their relationships on a 2D plane, to facilitate their management.

The authoring tool allow us to rapidly create simple 3D scenarios with active virtual humans. Inside the 3D scene, virtual humans are able to perform different actions, which are limited to those that the 3D engine can provide. However, the system is scalable thanks to the ontology. Each time a new functionality is available on the 3D engine, it can be described in the ontology and exposed through the visual programming interface. Moreover, using the ontology we are able to program virtual humans in a higher level of semantics, and as a consequence, we can focus on higher-level tasks.

Our system presents some challenges for the developers of 3D engines who may want to use it as authoring tool. It requires an understandable formalization of features, and ad hoc interfaces should be used (e.g., Protégé). Using ontology expressiveness to configure systems is a novel application of semantics. Ontology manipulation systems are not designed with virtual environments development in mind, thus interfacing the knowledge in the ontology with a VE application is no trivial task. However, once the ontology is set up, the application provides a user-friendly tool for the development of VR applications.

Future work includes addressing the problems cited before. We plan to develop new tools that automate the population of the ontology and its maintenance.

Acknowledgement We would like to thank Philippe Mazouer for his help in the development of this application. This research has been funded by the EC and the Swiss Federal Office for Education and Science in the framework of the European Network of Excellence IST-AIM@SHAPE (http://www.aimatshape.net).

References

1. Act-3D: Quest 3D. http://www.quest3d.com. Accessed (2006)
2. Celentano, A., Pittarello, F.: A content-centered methodology for authoring 3D interactive worlds for cultural heritage. In: ICHIM (2), pp. 315–324. Archives and Museum Informatics, Pittsburgh, PA (2001)
3. Cera, C., Regli, W., Braude, I., Shapirstein, Y., Foster, C.: A collaborative 3D environment for authoring design semantics. IEEE Comput. Graph. Appl. **22**(3), 43–55 (2002)
4. Costagliola, G., Martino, S.D., Ferrucci, F., Pittarello, F.: An approach for authoring 3d cultural heritage exhibitions on the web. In: SEKE '02: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, pp. 601–608. ACM, New York (2002). doi: 10.1145/568760.568865
5. EA Inc.: EA games. http://www.ea.com. Accessed (2007)
6. Green, M.: Towards virtual environment authoring tools for content developers. In: VRST '03: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 117–123. ACM, New York (2003). http://doi.acm.org/10.1145/1008653.1008675
7. Gutiérrez, M.: Semantic virtual environments. Dissertation, EPFL, Lausanne (2005)
8. Gutiérrez, M., García-Rojas, A., Thalmann, D., Vexo, F., Moccozet, L., Magnenat-Thalmann, N., Mortara, M., Spagnuolo, M.: An ontology of virtual humans: Incorporating semantics into human shapes. Visual Comput. **23**(3), 207–218 (2007)
9. Kalogerakis, E., Christodoulakis, S., Moumoutzis, N.: Coupling ontologies with graphics content for knowledge driven visualization. In: VR '06: Proceedings of the IEEE Virtual Reality Conference (VR 2006), p. 6. IEEE Computer Society, Washington, DC (2006). http://dx.doi.org/10.1109/VR.2006.41
10. Klesen, M., Kipp, M., Gebhard, P., Rist, T.: Staging exhibitions: methods and tools for modelling narrative structure to produce interactive performances with virtual actors. Virtual Reality **7**(1), 17–29 (2003)
11. McBride, B.: Jena: a semantic web toolkit. IEEE Internet Comput. **6**(6), 55–59 (2002). doi: 10.1109/MIC.2002.1067737
12. NetBeans: Visual library. http://www.w3.org/tr/owl-features/. Accessed (2007)
13. Perlin, K., Goldberg, A.: Improv: A system for scripting interactive actors in virtual worlds. Comput. Graph. **30**(Annual Conference Series), 205–216 (1996)

14. Peternier, A., Thalmann, D., Vexo, F.: Mental vision: a computer graphics teaching platform. In: Edutainment 2006. Lect. Notes Comput. Sci., vol. 3942, pp. 223–232. Springer, Berlin Heidelberg (2006)
15. Ponder, M., Papagiannakis, G., Molet, T., Magnenat-Thalmann, N., Thalmann, D.: VHD++ development framework: Towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies. In: Proceedings Computer Graphics International 2003, pp. 96–104. IEEE Computer Society, Los Alamitos, CA (2003)
16. Protégé: (c) 2005 Stanford Medical Informatics. http://protege.stanford.edu/index.html. Accessed (2007)
17. Virtools, S.: Virtools. http://www.virtools.com. Accessed (2006)
18. W3C: Owl Web Ontology Language. http://graph.netbeans.org. Accessed (2007)

ALEJANDRA GARCÍA-ROJAS received her Master's degree in Digital Interactive Technologies from the University Complutense of Madrid in 2004. She graduated as a Computer Scientist from Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Toluca in México. Since January 2005 she has been a Ph.D. candidate in The Virtual Reality Laboratory at the École Polytechnique Fédérale de Lausanne.

MARIO GUTIÉRREZ is a researcher at the Computer Science Department of the National Institute for Astrophysics, Optics and Electronics in México. He received his Ph.D. in Computer Science from the École Polytechnique Fédérale de Lausanne, Switzerland. His research interests include: semantic representations of virtual environments, human-computer interaction, robotics and virtual reality.

PROFESSOR DR. DANIEL THALMANN is a professor and Director of The Virtual Reality Lab (VRlab) at EPFL, Switzerland. He is a pioneer in research on virtual humans. His current research interests nclude real-time virtual humans in virtual reality, networked virtual environments, artificial life, and multimedia. Daniel Thalmann has been a professor at the University of Montreal and visiting professor/ researcher at CERN, University of Nebraska, University of Tokyo, and Institute of System Science in Singapore.
He is Co-Editor-in-Chief of the *Journal of Visualization and Computer Animation*, and member of the editorial board of *The Visual Computer* and four other journals. Daniel Thalmann was a member of numerous Program Committees, Program Chair and Co-Chair of several conferences. He is Program Co-Chair of CGI 2007, Virtual Rehabilitation 2007, ACM VRST 2008 and the Conference Co-Chair of CASA 2007 and SCA 2007. He has also organized five courses at SIGGRAPH on human animation. Daniel Thalmann has also been a member of Eurographics since 1980. He is a member of the Stearing Committee of the Symposium on Computer Animation, a joint Eurographics-SIGGRAPH initiative. Daniel Thalmann has published more than 400 papers in graphics, animation, and virtual reality. He is coeditor of 30 books, and coauthor of several books including the *Handbook on Virtual Humans*, published by Wiley. He was also co-director of several computer-generated films with synthetic actors including a synthetic Marilyn shown on numerous TV channels all over the world.
He received his Ph.D. in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul-Sabatier in Toulouse, France, in 2003.