A Virtual Human Architecture that Integrates Kinematic, Physical and Behavioral Aspects to Control H-Anim Characters

Lucio Ieronutti[1]
HCI Lab
Dept. of Math and Computer Science,
University of Udine
via delle Scienze, 206
33100 Udine, Italy

Luca Chittaro[2]
HCI Lab
Dept. of Math and Computer Science,
University of Udine
via delle Scienze, 206
33100 Udine, Italy

Abstract

Virtual humans are being increasingly used in different domains. Virtual human modeling requires to consider aspects belonging to different levels of abstractions. For example, at lower levels, one has to consider aspects concerning the geometric definition of the virtual human model and appearance while, at higher levels, one should be able to define how the virtual human behaves into an environment. H-Anim, the standard for representing humanoids in X3D/VRML worlds, is mainly concerned with low-level modeling aspects. As a result, the developer has to face the problem of defining the virtual human behavior and translating it into lower levels (e.g. geometrical and kinematic aspects). In this paper, we propose VHA (*Virtual Human Architecture*), a software architecture that allows one to easily manage an interactive H-Anim virtual human into X3D/VRML worlds. The proposed solution allows the developer to focus mainly on high-level aspects of the modeling process, such as the definition of the virtual human behavior.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques -- Interaction techniques. I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism -- Virtual reality. H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems -- Artificial, augmented, and virtual realities.

Keywords: H-Anim, Virtual Humans.

1. Introduction

Virtual humans, i.e. three-dimensional simulations of human beings, are being increasingly used in different domains. For example, they are used to explain physical and procedural human tasks [Badler et al. 2002; Rickel and Johnson 1999], they are employed in military applications [Traum and Rickel 2002], in ergonomics [Badler 1997], to simulate emergencies in first aid [Cassell et al. 1994], and as virtual guides [Chittaro et al. 2003; 2004].

Proper development of virtual humans requires knowledge of different disciplines, such as computational geometry, kinematics, artificial intelligence, computer graphics, and bio-mechanics. The complexity of building virtual humans encourages to divide the problem in several sub-problems; this can be done with the five-levels hierarchy proposed by [Funge et al. 1999], shown in Figure 1.

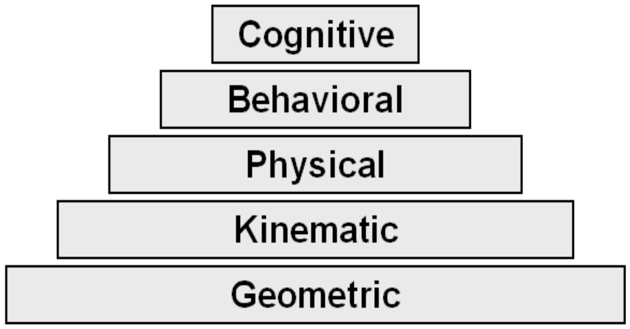


Figure 1. The modeling hierarchy proposed by [Funge et al. 1999].

The lowest layer of the modeling hierarchy is the *geometric layer* that concerns the definition of the virtual human model and its appearance. At the *kinematic layer*, the virtual human is represented as a set of rigid bodies, called *segments*, hierarchically organized and connected by *joints*. From this point of view, an animation can be defined in two ways: by specifying joints rotations, or by defining (or automatically computing) positions of specific parts of the virtual human body (called end-effectors) in time. The latter approach uses inverse kinematics to compute the joints configuration (in terms of rotation values) needed to put end-effectors in particular positions; this approach is commonly used to control hands and feet movements. At the *physical layer*, the animation is obtained by applying physical laws to different parts of the virtual human body to compute complex animations, such as skin deformation or hair movement. The *behavioral layer* represents the instinctive behavior of the virtual human (e.g. in terms of stimulus-action associations), while the highest layer, the *cognitive* one, binds various stimuli with reasoning processes that allow the virtual human to search for the most suitable action. Cognitive models go beyond behavioral models in that they govern what the virtual human knows, how that knowledge is acquired, and how it can be used to plan actions.

H-Anim [2004], the standard for representing humanoids in X3D/VRML worlds, deals only with lower layers of the modeling hierarchy. The standard defines the hierarchical structure of the virtual human in terms of `Segment` and `Joint nodes`, it defines

[1] e-mail: ieronutt@dimi.uniud.it
[2] e-mail: chittaro@dimi.uniud.it

the `Site node` to identify locations of semantic interest on the virtual human body (e.g. they can be used by inverse kinematics systems), and it uses the `Displacer node` to specify a particular mesh deformation (e.g. it can be used to produce facial expressions). On the other hand, H-Anim does not specify neither how to define the virtual human behavior nor how the different layers can be integrated.

As a result, when the developer wants to include an interactive virtual human into a X3D/VRML world, she has to write a considerable amount of code that is related to the different layers of the hierarchy. Moreover, most of this ad-hoc work has to be rethought if the developer needs to integrate the virtual human into another application. Therefore, the implementation of virtual humans is an inefficient and partly undisciplined activity for the developer.

In this paper, we present VHA (*Virtual Human Architecture*), a software architecture that allows one to develop interactive H-Anim virtual humans by clearly separating the geometrical, kinematic, physical and behavioral layers of the modeling hierarchy. The goal of our work is twofold. First, we intend to study a possible solution to overcome some modeling problems of H-Anim virtual humans. Second, we intend to propose an architecture that, although less sophisticated than other architectures, can be easily run on the Web using common PCs.

This paper is structured as follows. First, in Section 2, we discuss related work. Then, in Section 3, we describe the software architecture we propose. In Section 4, we illustrate some implementation choices we have taken for implementing the architecture. Section 5 outlines two case studies we used for testing our solution. Finally, in Section 6, we discuss the current limitations and future developments of the proposed architecture.

2. Related Work

Simulating human motion and behavior using a computer is an active research area (see, e.g. [Ponder et al. 2003; Badler et al. 2002]) and different approaches have been proposed in literature. Each approach considers the appearance, function and autonomy of the virtual human with respect to specific levels of detail and accuracy, and given application areas.

A commercial system that comprises many aspects of human motion and simulation is *Jack* [Badler et al. 1993], an interactive system for the definition, manipulation, animation, and performance analysis of virtual humans. Jack is suitable for ergonomics applications and contains kinematic and dynamic models of humans to simulate complex behaviors, such as balance, grasping and walking. A similar system is *HUMANOID* [Boulic et al. 1995], which includes modules to compute deformations of the skin, hand and face of the virtual human. *VLNET* [Pandzic et al. 1998] is a networked multi-user virtual environment that uses the HUMANOID articulated body model as a basis for virtual human display and motion.

Another interesting system is *STEVE* (Soar Training Expert for Virtual Environments) [Rickel and Johnson 1999]: STEVE can demonstrate skills to students, answer student questions, watch the students as they perform the tasks, and give advice if the students run into difficulties. However, the complex functionalities provided by STEVE require to run different modules of the system in parallel as separate processes, possibly on different computers.

Some systems allow one to define the behavior of virtual humans through scripting languages. In this context, an interesting system is *Improv* [Perlin and Goldberg 1996] that represents each virtual human behavior as a set of rules that specify how the virtual human communicates and takes decisions. However, Improv is more suitable for interactive storytelling rather than virtual environments. Another interesting system is *SimHuman* [Vosinakis and Panayiotopoulos 2001]: it simulates virtual humans with planning capabilities and is devoted to support real-time applications, taking into account requirements of natural looking motion and acceptable execution speed.

Another interesting system is *VHD++* [Ponder et al. 2003], a real-time software framework (written in C++) for developing virtual and augmented reality applications employing advanced virtual character simulation technologies. In particular, VHD++ is an efficient, flexible and extendible framework based on modern 3D game-engine design principles. VHD++ can be employed in a large number of different scenarios; it has been used i) in simulations of cultural heritage involving architecture, acoustics and cloths, ii) in augmented reality applications for maintenance training, iii) in edutainment systems based on storytelling and iv) in immersive applications designed to train non-professional health emergency operators.

3. The Proposed Virtual Human Architecture

VHA (*Virtual Human Architecture*) is a software architecture that allows one to develop interactive H-Anim virtual humans by clearly separating the geometrical, kinematic, physical and behavioral layers of the modeling hierarchy. In particular, VHA allows one to implement virtual humans whose behavior is based on the *Sense-Decide-Act* paradigm; the virtual human is able to perceive what happens in the surrounding environment (sense process), to decide a proper reaction (decide process) and to perform the related actions in the Web3D world (act process).

The main features of VHA are the following. First, the architecture is general, since it does not depend from a specific virtual human application and 3D world.

Second, it finds a good compromise between the required realism of the representation (at each level of the modeling hierarchy) and the efficiency of the simulation (the simulation has to be run in real-time on common PCs).

Third, unlike all systems described in Section 2, the VHA supports execution on the Web, since it is completely based on Web standards, such as X3D/VRML, H-Anim and Java.

To integrate an interactive H-Anim virtual human into a virtual environment, the VHA uses the following data: i) specification of how the virtual human should react to users' actions, ii) the virtual environment topology (e.g. navigable areas), iii) information about objects (e.g. position and geometry of an object), iv) a set of virtual human animations and v) the H-Anim model of the virtual human.

The main internal modules of the architecture (depicted in Figure 2) are the *Behavioral Engine*, the *Execution Engine* and the *Presentation Module*. The Behavioral Engine refers to the Behavioral layer of the modeling hierarchy (the sense and decide processes), the Execution Engine deals with the kinematic, physical and geometric layers (the act process), while the Presentation module is used by the system for presenting textual/vocal information to the user.

The Behavioral Engine senses user's actions and identifies the set of high-level actions the virtual human has to perform in response. These actions are sent to the Execution Engine, which both identifies the animations the virtual human has to perform and retrieves information to be provided to the user. Information is sent to the Presentation Module that transforms the textual data in a format suitable for the presentation.

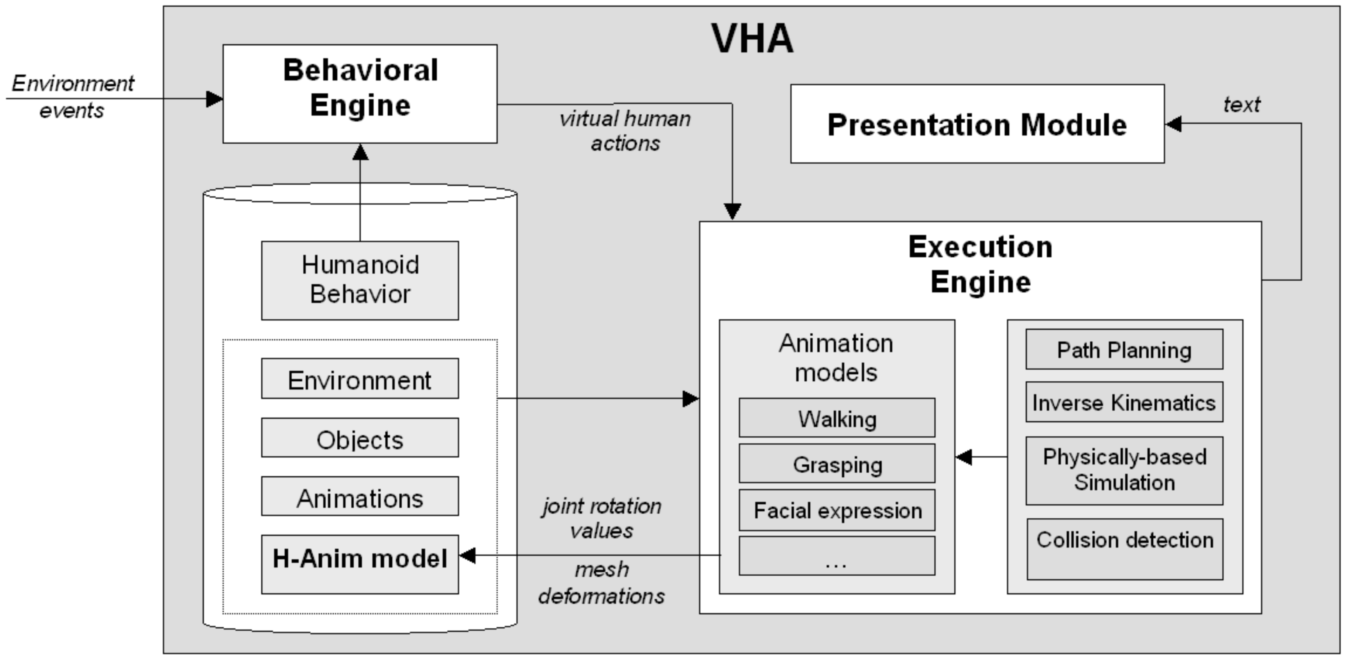


Figure 2. The proposed Virtual Human Architecture.

In the following sections, we describe each module of the proposed architecture in more detail.

3.1 The Behavioral Engine

The Behavioral Engine senses environment events and determines the proper virtual human actions. Environment events are stored into an internal buffer (called *Events Buffer*). The Behavioral Engine defines how the virtual human behavior in terms of Finite-State Machines (FSMs). In particular, FSMs allow the developer to specify how the virtual human behaves, given its current internal state and by considering events stored into the Events Buffer.

A FSM is represented by a directed graph $G = (V, E)$, where V represents a set of nodes n_i, and E represents a set of oriented edges (n_i, n_j). Each node corresponds to a particular state of the virtual human, while each edge corresponds to a transition that allows the virtual human to change its internal state. Conventionally the node n_0 represents the initial state, and nodes that do not have outgoing edges are called *end states*.

Each transition (n_i, n_j) is characterized by conditions-actions (c_{ij}, a_{ij}) pairs: c_{ij} is the set of conditions that determine the applicability of the transition, while a_{ij} is the set of actions (animation performed and information presented) the virtual human has to execute if the corresponding transition is activated.

Figure 3 shows a FSM that describes a very simple virtual human behavior. The FSM is represented by three states (n_0, n_1 and n_2) and four transitions ((n_0, n_1), (n_0, n_2), (n_2, n_2) and (n_2, n_1)), each one characterized by one conditions-actions (c_{ij}, a_{ij}) pair.

The correspondent virtual human behavior is the following. If both the user and the virtual human are close to an object called `object_1` (c_{01}), the virtual human presents the object (a_{01}). On the other hand, if they are far from the object (c_{02}), the virtual human firstly invites the user to follow it, and then it walks to the object (a_{02}). The virtual human waits (a_{22}) until the user is close enough to the object (c_{22}); when this happens, i.e. the user is near the object (c_{21}), the virtual human presents it (a_{21}).

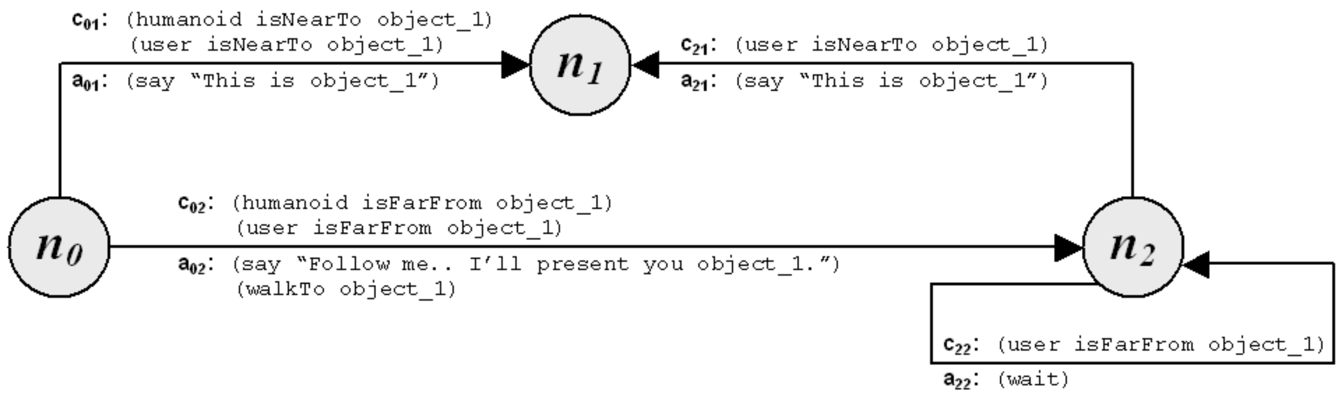

c01: (humanoid isNearTo object_1)
(user isNearTo object_1)
a01: (say "This is object_1")
n1
c21: (user isNearTo object_1)
a21: (say "This is object_1")
c02: (humanoid isFarFrom object_1)
(user isFarFrom object_1)
n0
a02: (say "Follow me.. I'll present you object_1.")
(walkTo object_1)
n2
c22: (user isFarFrom object_1)
a22: (wait)

Figure 3. An example of a simple Finite-State Machine (FSM).

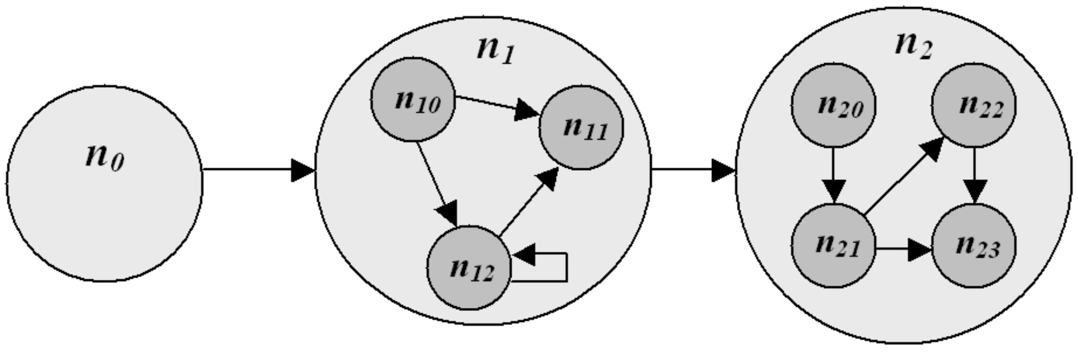


Figure 4. An example of a Hierarchical-State Machine (HSM)

A transition *(n_i, n_j)* can be activated by the Behavioral Engine if and only if all corresponding conditions c_{ij} are satisfied; a single condition of c_{ij} is considered satisfied when it matches with an event stored into the Events Buffer. Conditions can concern events generated by explicit user actions (e.g. an object has been touched by the user), they can refer to spatial relations (e.g. the user is close to the virtual human) or temporal conditions (e.g. the user does not interact with any object for some time). The decision process is activated whenever all actions of the previously activated transition have been executed by the system; in other words, in our model the execution of a set of actions a_{ij} is not interruptible.

Once a transition *(n_i, n_j)* is activated, the Behavioral Engine deletes all events stored into the Events Buffer, sends to the Execution Engine the list of actions a_{ij}, and then starts to sense new events.

VHA allows one to describe complex virtual human behavior through Hierarchical-State Machines (HSMs), i.e. hierarchical compositions of FSMs. Each graph node can be either a *basic state* (i.e. it belongs to V) or a state that contains the description of a particular behavior (represented through a HSM itself). In our system, a transition can be activated if and only if the current virtual human state is an end state belonging to the underlying level (e.g. in Figure 4 the transition *(n_1, n_2)* can be activated if the current virtual human state is n_{11}). The hierarchy is multi-level, since each HSM can contain nodes that do not represent basic states. For a formal treatment of HSMs, readers can refer e.g. to Mikk [Mikk et al. 1997].

Figure 4 shows the typical structure of a HSM; in this example, the HSM is composed by two levels and represents a simple virtual guide behavior (e.g. the virtual human leads the user through the environment by presenting sequentially two objects). The top layer is composed by three nodes (n_0, n_1 and n_2) and represents the high-level description of the virtual guide behavior. In particular, node n_0 represents the initial state of the virtual guide, while n_1 and n_2 correspond to the presentation of two different objects. The lower level is represented by two HSMs corresponding to descriptions of more specific virtual guide behaviors. For example, the developer can place into the node n_1 the FSM depicted in Figure 3 to describe the particular virtual human behavior corresponding to `object_1` presentation.

Given a current state of the virtual human, the Behavioral Engine senses environment events, determines what conditions are satisfied, identifies applicable transitions, activates one of them and sends to the Execution Engine the ordered list of actions associated with the chosen transition. Actions correspond to animations that the virtual human has to perform and information that has to be presented to the user.

3.2 The Execution Engine

To define an animation for H-Anim virtual humans, the developer has to specify different joint rotation values over time; the resulting virtual human motion is generated by smoothly interpolating specified rotation values. Similarly, to define a mesh deformation (e.g. the skin of the virtual human face is modified to obtain a facial expression), she has to specify different positions of mesh vertexes over time; the resulting deformation is obtained by interpolating specified position values. This kind of animations are called *pre-stored animations*, since the complete description of the movement has to be specified in advance.

Another kind of animation, called *parametrized animations*, is more convenient to animate a virtual human. Parametrized animations are more general and flexible than pre-stored ones, since they allow one to generate a variety of movements only by changing a small set of animation parameters. Parametrized animations are usually based on inverse kinematics to control end-effectors movements (e.g. virtual human feet and hands) and employ path planning algorithms to generate collision-free motions.

The Execution Engine is able both to use pre-stored animations (e.g. recorded by using Motion Capture devices) and to run parametrized animations at execution-time. Each parametrized animation is represented by an *animation model* that describes how the animation is generated starting from a set of given parameters. An animation model can use information on the topology of the environment (e.g. areas that can be traveled by the virtual human), object information (e.g. object position and geometry) and parameters explicitly given by the developer (e.g. a command (walk`To coord 5 2 4)`) to generate the virtual human animation.

3.3 The Presentation Module

The Presentation Module allows the virtual human to present textual information to the user. The required information can be shown on a 2D On-Screen Display (OSD) into the virtual environment (Figure 5) and/or presented by using a synthesized voice.

The Presentation Module can format the given textual information to adapt the visualization according to the display dimension and, at the same time, can control the synthesized voice volume by taking into account the distance between the virtual human and the user.

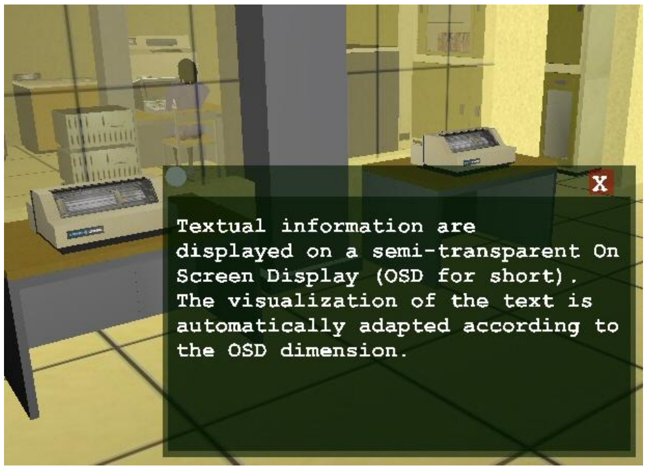

Textual information are displayed on a semi-transparent On Screen Display (OSD for short). The visualization of the text is automatically adapted according to the OSD dimension.

Figure 5. A textual information displayed on the OSD.

4. VHA Implementation

In this Section, we provide some implementation details such as the models used for generating parametrized animations and the algorithms employed for the Path Planning, the Inverse Kinematics, the Physically-based Simulation and the Collision Detection modules (shown in Figure 2). Obviously, different approaches could be followed to implement the same modules without changing the general architecture.

Our implementation choices are described in the following Section, while in Section 4.2 we present how we integrate VHA into a generic VRML environment.

4.1 Animation models

The Execution Engine we implemented contains different animation models, but the most relevant ones are the grasp model, the walk model, and the model that allows the virtual human to perform facial expressions (Figure 6 shows examples of animations produced by the three models). Animation models use pre-stored data, the Inverse Kinematics, the Path Planning, the Collision Detection and the Physically-based simulation modules to generate the motion.

The grasp model we implemented employs the *Multi-sensor* approach [Huang et al. 1995] to simplify the collision-detection algorithm. This technique hangs spherical sensors to the articulated figure and activates each sensor for any collision (detected by the Collision Detection module) with other objects or sensors. The Inverse Kinematics module is used by the grasp model to compute joint rotation values suitable to place the virtual human hands in the required positions.

The walk model we implemented allows the virtual human to walk on both even and uneven terrains. The walk model combines the automatic generation of human walking motion (based on the model proposed by [Chung and Hahn 1999]) with trajectory information (automatically derived by the implemented Path Planning module). In particular, the walk model first uses both the trajectory information and the data representing the walking surface topology to derive foot and pelvis trajectories. Then, it employs the Inverse Kinematics module to compute proper joint rotation values for the virtual human legs. Finally, the walk model employs generic pre-stored data to animate the upper part of the virtual human body.

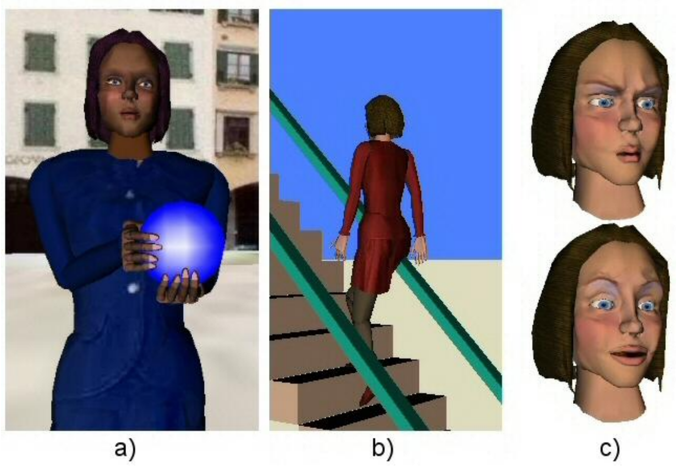


Figure 6. Example of parametrized animations supported by VHA: a) grasping, b) walking on uneven surfaces and c) facial expressions.

The animation model for facial expressions employs the *muscle-based* approach [Lee et al. 1995] that controls facial expressions by acting on muscle contractions. In particular, the model we implemented represents the virtual human face as a deformable multi-layered mesh; nodes of the mesh are considered mass elements connected by spring elements. Nodes are arranged in three layers: the top layer represents the epidermis, the middle layer represents the tissue, and the bottom layer represents the skull surface. The elements between the top and middle layers represent the fatty tissues, while the elements between the middle and bottom layer represent the facial muscles. The mesh is driven by modeling the activation and motion of several facial muscles in various facial expressions. The Physically-based Simulation module is used to simulate the skin behavior.

In the following, we give an overview of the techniques we employed for implementing the Path Planning, the Inverse Kinematics and the Physically-based Simulation modules.

Path Planning. Many algorithms have been proposed for path planning (a good survey of classical methods is presented in [Latombe 1991]). The approach we implemented for the Path Planning module captures the topology of the configuration space (i.e. collision-free positions) by representing it into a graph structure. The Path Planning technique we use is an extended version of the approach we presented in [Chittaro et al. 2003]. This new version computes a collision-free path also in multi-floor environments. Each floor of the virtual environment is represented as an *Occupancy Grid*, a two dimensional matrix such that: (i) each cell of the matrix corresponds to an area of the floor and, (ii) the value of the cell indicates whether the corresponding area contains a geometry or not. The Path Planning module derives from the Occupancy Grids a graph that captures the topology of the environment. This structure allows the Path Planning module to compute collision-free paths by employing a traditional searching algorithm (e.g. Dijkstra or A*). To automatically derive Occupancy Grids starting from a X3D/VRML world, we employ the technique we proposed in [Ieronutti et al. 2004]. The basic idea of this technique is to determine whether a cell of the floor should indicate the presence of geometry that prevents navigation, by checking if the corresponding area in the world can be traveled by an avatar created for the purpose.

Inverse Kinematics. Inverse kinematics can use different methods to solve the problem of finding the joints configuration that allows the end-effector to reach the desired position. Two main categories of solutions can be identified in the literature: closed form solutions and numerical solutions [Lucas et al. 2000]. The first category of approaches analytically computes a solution by using non-iterative calculations. In general, these systems employ algebraic and geometric techniques to compute a solution quickly and precisely. On the other hand, it is better to choose numerical solutions when the system is too complicated for the closed form methods; numerical solutions use iterative calculations to approach a solution as closely as possible, requiring a longer time. However, this type of approaches can solve also very complex kinematic systems. An alternative technique, based on neural networks, has been proposed to solve inverse kinematic systems [de Lope et al. 2003]. The Inverse Kinematics module we implemented belongs to the first category of solutions; it solves the problem of moving limbs composed of two segments in a three-dimensional space.

Physically-based Simulation. The Physically-based simulation module we implemented allows VHA to simulate elastic surfaces through a mass-spring system. The dynamic behavior of a surface is computed by numerically integrating positions and velocities of mass elements over time. The computation cost of the simulation is mainly due to numerical integration of the ordinary differential equation systems that model the deformable surface. The simulation engine we employ has been described in detail in [Chittaro and Corvaglia 2003] and implements three explicit integration methods, while implicit methods have not been considered (to limit simulation complexity).

4.2 VRML nodes introduced for using VHA

From a VRML point of view, VHA is seen as a `PROTO` (row 1 in the listed code below). Whenever the developer intends to include the VHA into a virtual environment, she has to instantiate the `VHA` node by specifying the virtual human behavior (row 3), environment information (row 4), object information (row 5), the set of pre-stored animations the virtual human is able to perform (row 6) and the H-Anim model representing the virtual human (row 7).

VHA senses environment events (e.g. the user interacts with a `TouchSensor` node) through the `VeEvent eventIn` (row 2). The developer can choose arbitrarily the syntax used for representing environment events, provided that she follows the same syntax in the HSMs conditions.

In particular, the `behaviors, objects and animations` of the `VHA` fields contain respectively a list of `Behavior, Object and Animation` nodes. Moreover, the first element of the `behaviors` field represents the top layer of the HSM, while the other elements (representing HSMs belonging to lower layers) are loaded only when needed by using a command `(import behavior` *`behavior_Name`*`)`.

```
1. PROTO VHA [
2.   eventIn SFString VeEvent
3.   exposedField MFNode behaviors []
4.   exposedField SFNode environment
       Environment{}
5.   exposedField MFNode objects []
6.   exposedField MFNode animations []
7.   exposedField SFNode humanoid Humanoid{}
8. ]{… }
```

In the following sections, we describe the VRML `PROTO`s introduced for integrating the VHA into VRML words, by presenting their interfaces.

4.2.1 Behavior node

Each virtual human behavior is represented by a `Behavior PROTO` (row 9). The node is characterized by the name of the behavior (row 10) and contains information on HSM transitions (row 11); each transition (row 13) is characterized by a set of conditions (row 14) and actions (row 15), both represented by using a string array. Each single string represents a different condition or action, while empty spaces are used to separate different parameters.

```
9. PROTO Behavior [
10.   exposedField SFString name ""
11.   exposedField MFNode transitions []
12. ]{ }

13. PROTO Transition [
14.   exposedField MFString conditions ""
15.   exposedField MFString actions ""
16. ]{ }
```

4.2.2 Environment node

The `Environment PROTO` (row 17) stores information on the topology of the environment; the VHA uses this information to control virtual human movements through multi-floor virtual environments.

In particular, the `Environment PROTO` is characterized by the address of the file containing the virtual environment (row 18), the set of parameters used for deriving obstacles information (rows 19-23), floor information (row 24) and information on the stairs connecting different floors (row 25). `xLen`, `zLen`, `accuracy`, `xPos` and `zPos` (rows 19-23) represent parameters used for the automatic derivation of the `occupancyGrid` (see [Ieronutti et al. 2004] for the precise meaning of these parameters). The `floors` (row 24) and `stairs` (row 25) fields of the `Environment` node contains respectively a list of `Floor` and `Stair` nodes.

```
17. PROTO Environment [
18.   exposedField SFString url ""
19.   exposedField SFInt32 xLen 0
20.   exposedField SFInt32 zLen 0
21.   exposedField SFInt32 accuracy 1
22.   exposedField SFInt32 xPos
23.   exposedField SFInt32 zPos
24.   exposedField MFNode  floors [ ]
25.   exposedField MFNode  stairs [ ]
26. ]{ }
```

The `Floor PROTO` (row 27) contains information on the surface on which the virtual human has to walk; this information is represented by a pair of two-dimensional matrices, the `occupancyGrid` (row 28) and the `heightGrid` (row 29). The `occupancyGrid` corresponds to the Occupancy Grid described in Section 4.1; this information is represented as a string in which the empty character and the character '`X`' are used for identifying respectively travelable areas and areas that contain an obstacle to the virtual human navigation. The `heightGrid` is a matrix having the same dimensions of the `occupancyGrid`, but in which the value of each cell specifies the height of the surface above the corresponding area (like the field `height` of the X3D/VRML `ElevationGrid` node). If the floor surface is completely flat, the `heightGrid` can be represented by an integer that specify the vertical position of the floor surface with respect to the absolute coordinate system.

```
27. PROTO Floor [
28.   exposedField SFString occupancyGrid ""
29.   exposedField MFFloat heightGrid 0
30. ]{ }
```

Different floors can be connected by stairs (row 31); each stair is defined by its position (row 32) and by a pair of integers that specifies which floors the stair connects (`floorL` stands for lower floor while `floorU` stands for upper floor, rows 33-34). The other three parameters (rows 35-37) define other characteristics of the stair; the first one represents the number of steps of the stair and the other two parameters represents respectively the height and the length of a single step.

```
31. PROTO Stair [
32.   exposedField SFVec3f position 0 0 0
33.   exposedField SFInt32 floorL 0
34.   exposedField SFInt32 floorU 0
35.   exposedField SFInt32 numSteps 0
36.   exposedField SFFloat stepHeight 0
37.   exposedField SFFloat stepLength 0
38. ]{ }
```

4.2.3 Object node

Object information is represented by the `Object PROTO` (row 39). In particular, the node specifies the name of the object (row 40), its position and orientation (rows 41-42), textual information (row 43) and the definition of the object geometry (row 44).

VHA uses object names for identification purposes; this way, actions can use the name of an object as a parameter (e.g. `goTo object printer`). The object position and orientation are used both to position the object into the virtual environment and to identify a position of semantic interest (e.g. the above parameter `printer` is automatically associated to the corresponding position). We have included in `Object` node the `description` field since in several contexts it can be useful to associate textual information to different objects e.g. the description of an object can be presented by the virtual human by simply using the action `(describe` *`object_Name`*`)`.

```
39. PROTO Object [
40.   exposedField SFString name ""
41.   exposedField SFVec3f position 0 0 0
42.   exposedField SFRotation orientation 0 1 0 0
43.   exposedField SFString description ""
44.   field MFNode shape NULL
45. ]{ }
```

4.2.4 Animation node

The VHA supports both pre-stored animations and parametrized ones. Pre-stored animations are represented through the `Animation PROTO` (row 46). A pre-stored animation is characterized by its name (row 47), duration (row 48), the number of times the animation has to be repeated (row 49) and the `Interpolators` node (row 50). The `Interpolators PROTO` (row 54) specifies both the `key` and `keyValue` fields for a list of `OrientationInterpolators`, each one related to a different (H-Anim) joint involved in the motion (the `PROTO` includes a field for each H-Anim joint).

The `Animation` node we implemented provides the possibility to organize different animations in groups; each animation can belong to a group of animations, whose name is defined by the field `group` (row 52). The `probability` field (row 53) represents the probability the animation is chosen by the Execution Engine when it needs to retrieve an animation belonging to a specific group.

```
46. PROTO Animation [
47.   exposedField SFString name ""
48.   exposedField SFTime duration 0
49.   exposedField SFInt32 loop 1
50.   exposedField SFNode  interpolators
51.                               Interpolators{}
52.   exposedField SFString group ""
53.   exposedField SFFloat probability 1
54. ]{ }

55. PROTO Interpolators [
56.   exposedField MFVec3f HumanoidRoot_KV []
57.   exposedField MFFloat HumanoidRoot_K []
58.   exposedField MFVec3f Sacroiliac_KV []
59.   exposedField MFVec3f …
      …
60. ]{ }
```

The organization into groups of animations allows the system to improve the variety of the virtual human motion. For example, suppose that the developer has defined several waiting animations (e.g. the virtual human looks around, scratches its head, breaths, …) and the virtual human is waiting for user action. The system is able to automatically retrieve a waiting animation by considering the probability associated to each single animation that belongs to the required group (e.g. it is more probable that the virtual human breaths rather than scratching its head). Moreover, given a wide number of pre-stored animations, this approach allows the developer to modify how the virtual human behaves only by changing the `probability` value of different animations.

4.2.5 Presentation node

The `Presentation PROTO` (row 61) implements the Presentation Module (described in Section 3.3); in particular, the node is used by `VHA` for presenting information to the user.

Through the `Presentation` node, textual information is at the same time displayed on a OSD and presented by using a synthesized voice (e.g. by using Microsoft TTs engine).

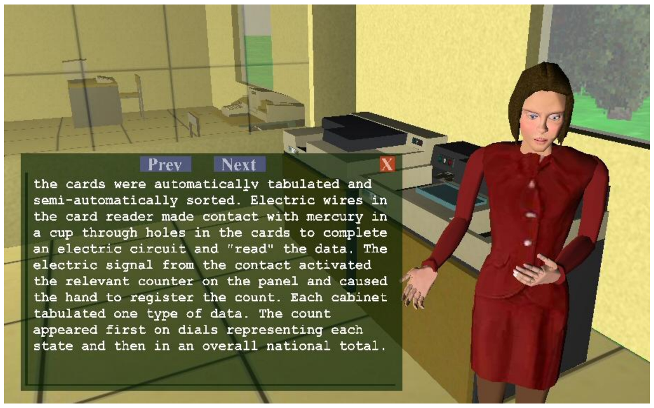
The implemented OSD follows the user, and then requires the position and orientation of the user (rows 62-63) to update its position during user's movements. Given the text to be displayed (row 64), the node adapts the corresponding string visualization according to the display dimension (defined in rows 66-67).

```
61. PROTO Presentation [
62.     eventIn  SFVec3f userPosition
63.     eventIn  SFRotation userRotation
64.     eventIn  SFString text
65.     eventIn  SFFloat distance
66.     field SFFloat xDim
67.     field SFFloat yDim
68. ]{…}
```

To present information by using a synthesized voice, the `Presentation PROTO` sends to a JavaScript (inserted into an hidden HTML frame) the text that has to be spoken (row 64). Moreover, given the distance between the user and the virtual human (row 65), the `Presentation PROTO` controls the synthesized voice volume.

5. Case Studies

The proposed architecture can be easily integrated into a generic X3D/VRML world. To test the effectiveness of VHA, we considered two different case studies. First, we employed a virtual human into a 3D Computer Science museum based on a VRML world representing a data processing center of the 70's (see Figure 7). Second, we used the same virtual human as a guide into an architectural virtual reconstruction of cultural heritage (see Figure 8). Although the considered Web3D sites differ in purpose and content, only a few minor modifications (e.g. the description of the virtual human behavior and environment information) have been necessary to move from the first to the second application.

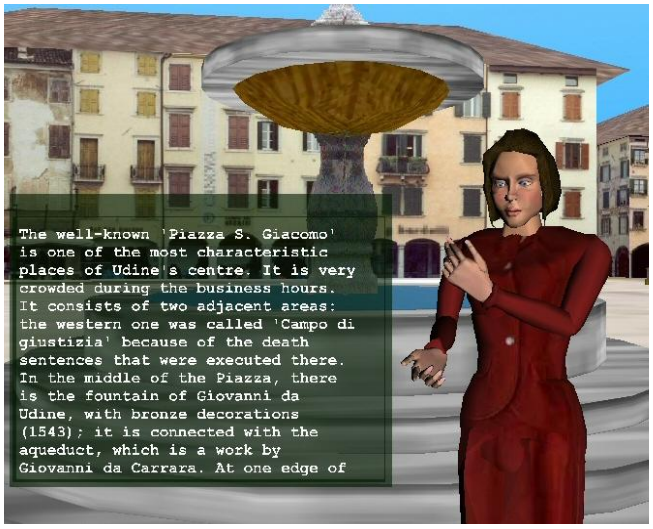


```
the cards were automatically tabulated and
semi-automatically sorted. Electric wires in
the card reader made contact with mercury in
a cup through holes in the cards to complete
an electric circuit and "read" the data. The
electric signal from the contact activated
the relevant counter on the panel and caused
the hand to register the count. Each cabinet
tabulated one type of data. The count
appeared first on dials representing each
state and then in an overall national total.
```

Figure 7. The virtual human explaining the functioning of the card punch in the Computer Science museum.

While in the first environment the virtual human is able to provide technical information by demonstrating how different devices worked, in the second application the same virtual human is used to tell the story of different buildings and highlight the main architectural differences.

In particular, for the first application, we have defined different HSM corresponding to different virtual human behaviors, each one designed to highlight a different aspect of the Computer Science museum (e.g. an high-level introduction to the overall environment, an explanation of the hardware architecture, a description of work activities). The user, according to her personal interests and needs, can choose (and change during the visit) the topics the virtual human has to explain.



```
The well-known 'Piazza S. Giacomo'
is one of the most characteristic
places of Udine's centre. It is very
crowded during the business hours.
It consists of two adjacent areas:
the western one was called 'Campo di
giustizia' because of the death
sentences that were executed there.
In the middle of the Piazza, there
is the fountain of Giovanni da
Udine, with bronze decorations
(1543); it is connected with the
aqueduct, which is a work by
Giovanni da Carrara. At one edge of
```

Figure 8. The virtual human providing architectural information.

Recently we have evaluated the positive effects of using virtual humans in the context of virtual museums and in making a virtual place more lively and attractive for users [Chittaro et al 2004]. As a navigation aid, the virtual human proved to be appreciated in the evaluation; it was mostly rated as simple to use, and it had the advantage of being unobtrusive for the expert user. Moreover, using the virtual human demanded a very short learning time, probably because, from a human-computer interaction point of view, the virtual human metaphor has the advantage of being consistent with the real-world experience of users. On the other hand, by analyzing the most frequent concerns expressed by subjects, there is a clear need for personalization capabilities. For example, walking and talking speed of the virtual human were both rated too slow or too fast by some users: the ideal solution would be to adapt these features on the basis of each single user's preference.

6. Conclusions

This paper proposed VHA, an architecture that integrates the kinematic, physical and behavioral aspects to control H-Anim virtual humans. The proposed solution, fully compatible with Web standards, allows the developer to easily augment X3D/VRML worlds with interactive H-Anim virtual humans whose behavior is based on the Sense-Decide-Act paradigm (represented through HSMs).

With respect to future goals of this project, we plan to extend VHA in order to support more than one H-Anim virtual human at the same time, and allowing virtual humans to interact with each other. Moreover, we intend to improve the capability of generating parametrized animations. From this point of view, we intend to integrate into the architecture two additional modules: a module to synchronize lip movements with spoken information and one to consider reachability problems for the generation of the grasping movement. Furthermore, since the time required for describing complex virtual human behavior (e.g. when the HSM is represented by more than 30 states) can be considerable, we plan to develop an authoring tool that allows the Web3D content creator to define the virtual human behavior either through a graphical user interface or by employing a more abstract programming language (e.g. by using the Virtual Human Markup Language [VHML 2004]).

Finally, we plan to develop the cognitive layer into the proposed architecture; reasoning processes should allow to control what the virtual human knows, how that knowledge is acquired, and how it can be used both to plan actions and to generate virtual human emotions. In this context, we have separately studied [Chittaro and Serra 2004] a system for modeling personality aspects into H-Anim virtual humans. To develop a *Cognitive Module* for VHA, we have started using the Java Expert System Shell [Jess 2004], that allows us to simulate reasoning processes by employing a knowledge base and a set of declarative rules.

7. References

BADLER N. 1997. Virtual Humans for Animation, Ergonomics, and Simulation. In *Proceedings of NAM '97: IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, IEEE Computer Society, USA.

BADLER, N., ERIGNAC, C., and LIU, Y. 2002. Virtual humans for validating maintenance procedures. *Communications of the ACM*, 45(7), 56-63.

BADLER, N., PHILLIPS, C., WEBBER, B. 1993. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press.

BOULIC, R., HUANG, Z., SHEN, J., MOLET, T., CAPIN, T., LINTERMANN, B., SAAR, K., THALMANN, D., MAGNETAT-THALMANN, N., SCHMITT, A., MOCCOZET, L., KALRA, P., and PANDZIC, I. 1995. A system for the parallel integrated motion of multiple deformable human characters with collision detection. In *Proceeding of EUROGRAPHICS'95*, 14(3), 337-348.

CASSELL, J., PELACHAUD, C., BADLER, N., STEEDMAN, M., ACHORN, B., BECKET, T., DOUVILLE, B., PREVOST, S., and STONE, M. 1994. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Proceedings of SIGGRAPH '94*, ACM Press, New York, 413-420.

CHITTARO L., and CORVAGLIA D. 2003. 3D Virtual Clothing: from Garment Design to Web3D Visualization and Simulation. In *Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology*, ACM Press, New York, 73-84.

CHITTARO, L., IERONUTTI, L., and RANON, R. 2004. Navigating 3D Virtual Environments by Following Embodied Agents: a Proposal and its Informal Evaluation on a Virtual Museum Application. *Psychology Journal* (Special issue on Human-Computer Interaction), *2*(1), 24-42.

CHITTARO L., RANON R., and IERONUTTI L. 2003. Guiding Visitors of Web3D Worlds through Automatically Generated Tours. In *Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology*, ACM Press, New York,.27-38.

CHITTARO L., and SERRA M. 2004. Behavioral Programming of Autonomous Characters based on Probabilistic Automata and Personality. *Journal of Computer Animation and Virtual Worlds*, *15* (3-4), 319-326.

CHUNG, S., and HAHN, J. K. 1999. Animation of Human Walking in Virtual Environments. In *Proceedings of Computer Animation*, IEEE Computer Society Press, 4-15.

DE LOPE, J., GONZÁLEZ-CAREAGA, R., ZARRAONANDIA, T., and MARAVALL, D. 2003. Inverse Kinematics for Humanoid Robots Using Artificial Neural Networks. In *Proceedings of EUROCAST: 8th International Conference on Computer Aided Systems Theory*, Springer-Verlag, Berlin, 448-459.

FUNGE, J., TU, X., and TERZOPOULOS, D. 1999. Cognitive Modeling: Knowledge, Reasoning, and Planning for Intelligent Characters. In *Proceedings of SIGGRAPH '99*, ACM Press, New York, 29-38.

H-ANIM WEB SITE. www.h-anim.org (last access on January 2004).

HUANG, Z., BOULIC, R., MAGNENAT-THALMANN, N., and THALMANN., D. 1995. A Multi-Sensor Approach for Grasping and 3D Interaction. In *Proceedings of CGI '95*, Academic Press, 235-253.

IERONUTTI L., RANON R., and CHITTARO L. 2004. Automatic Derivation of Electronic Maps from X3D/VRML Worlds. In *Proceedings of Web3D 2004: 9th International Conference on 3D Web Technology*, ACM Press, New York, 61-70.

JESS WEB SITE. http://herzberg.ca.sandia.gov/jess/ (last access on January 2004).

LATOMBE, J-C. 1991. *Robot Motion Planning*, Kluwer.

LEE, Y., TERZOPOULOS, D., and WATERS, K. 1995. Realistic Face Modeling for Animation. In *Proceedings of SIGGRAPH '95*, ACM Press, New York, 55-62.

LUCAS S. R., TISCHLER C. R., and SAMUEL A. E. 2000. Real-Time Solution of the Inverse Kinematic-Rate Problem. *International Journal of Robotics Research*, vol. 19, no. 12, 1236-1244.

MIKK, E., LAKHNECH, Y., and SIEGEL, M. 1997. Hierarchical Automata as Model for Statecharts. In *Proceedings of ASIAN '97: 3rd Asian Computer Science Conference, LNCS 1345*, Springer-Verlag, Berlin, 181-196.

PANDZIC, I.S., CAPIN, T.S., LEE, E., MAGNENAT-THALAMANN, N., and THALMANN, D. 1998. Autonomous Actors in Networked Collaborative Virtual Environments. In *Proceedings of MultiMedia Modeling '98*, IEEE Computer Society Press, 138-145.

PERLIN K., and GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH '96*, ACM Press, New York, 205-216.

PONDER, M., PAPAGIANNAKIS, G., MOLET, T., MAGNENAT-THALMANN, N., and THALMANN, D. 2003. VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies. *Proceedings of CGI'03*, IEEE Computer Society Press, 96-104.

RICKEL, J., and JOHNSON, W. L. 1999. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *Applied Artificial Intelligence* 13, 343-382.

TRAUM, D., and RICKEL, J. 2002. Embodied agents for multi-party dialogue in immersive virtual worlds. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, ACM Press, New York, 766-773.

VHML WEB SITE. http://www.vhml.org/ (last access on January 2004).

VOSINAKIS, S., and PANAYIOTOPOULOS, T. 2001. SimHuman: A Platform for Real-Time Virtual Agents with Planning capabilities. In *Proceedings of IVA '01 : 3rd International Workshop on Intelligent Virtual Agents*, Springer-Verlag, Berlin, 210-223.