

# SimHuman: A Platform for Real-Time Virtual Agents with Planning Capabilities

Spyros Vosinakis and Themis Panayiotopoulos

Knowledge Engineering Laboratory, Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou str., 18534, Piraeus, Greece  
spyrosv@unipi.gr, themisp@unipi.gr

**Abstract.** In this paper, we present SimHuman, a platform for the generation of real-time 3D environments with virtual agents. SimHuman is highly dynamic and configurable, as it is not based on fixed scenes and models, and has an embedded physically based modelling engine. Its agents can use features such as path finding, inverse kinematics and planning to achieve their goals. The paper explains in detail design and implementation issues and presents the architecture of the system as well as an illustrative example.

**Keywords:** virtual agents, believable agents, autonomous virtual humans, virtual environments, real-time animation, planning, simulation

## 1 Introduction

A virtual agent can be defined as an autonomous entity in a virtual environment. It should not only look like, but also behave as a living organism (human, animal or other fictional character) in a synthetic three-dimensional world, and be able to interact with it and its inhabitants. These could be either real users in the form of avatars, or other virtual agents.

There are numerous applications that would require some form of virtual agents in their environments, especially in fields such as entertainment, education, virtual environments, simulation, etc. [5]. There is, nevertheless, no standard definition of how such an ‘entity’ is to be implemented and this is due to the fact that different applications focus on different characteristics of such synthetic creatures. A game, for example, focuses mainly on appearance and behaviour, because the agents should be believable, while a simulation system aims at higher accuracy.

In this paper we present a system called SimHuman, which is our approach towards the generation of real-time virtual environments with intelligent virtual agents for desktop VR applications. This system can use any 3D model for agents, avatars and other virtual objects and is therefore highly dynamic and configurable. It has an embedded physically based modelling engine, and its agents can use features such as path finding, inverse kinematics and planning to achieve their goals. The system is designed in such a way that it maintains a balance between performance, autonomy and believability and can be used as a basis for real-time virtual environments and simple simulation systems.

The paper is structured as follows: In section 2 we present the related work in the field of virtual agents. The design issues of simulated humans for virtual environments are discussed in section 3, while the architecture of the SimHuman system is the subject of section 4. The next section presents an illustrative example of an application generated with SimHuman. Section 6 is concerned with the implementation issues, and in the last one we state our conclusions and possible extensions of this system.

## 2 Background

There has been a great amount of research in the field of virtual agent display, motion and behaviour, and a number of different systems and approaches have been proposed. Each of them varies in appearance, function and autonomy according to the application field and the required detail and accuracy.

The process of displaying and animating a virtual human involves three different stages. The most primitive one is the visualisation of the body, which is the same process as modelling any other 3D object. The next stage is the modelling of the skeleton, which defines the moving parts of the body and the type of motion that they can perform, and the last one is the modelling of skin and clothes, so as to move in a natural way [12]. The calculation of the skin and cloth motion is the most computationally intensive task, which is why it is not suitable for real-time animation.

Believable human body animation is a surprisingly hard task. The animation techniques fall into three basic categories: *keyframing*, *motion capture* and *simulation* [8]. All three involve a trade-off between the level of control that the animator has over the fine details of the motion and the amount of work that the computer does on its own.

In keyframing, the animator has to specify critical, or key, positions for the body parts, and the computer fills in the missing frames by smoothly interpolating between those positions. Body postures can be defined either with the low-level *forward kinematics* approach, or with the more elegant *inverse kinematics* one [29]. On the other hand, motion capture involves measuring a real person's / object's position and orientation in physical space, and then recording that information in a computer-usable form. [23].

Unlike keyframing and motion capture, simulation uses the laws of physics to generate motion of figures and other objects. Virtual humans are usually represented as a collection of rigid body parts. Although the models can be physically plausible, they are nonetheless only an approximation of the human body, because they ignore the movement of muscle mass relative to bone. Recently, researchers have begun to build more complex physical models based on bio-mechanical data, and the resulting simulations are becoming increasingly lifelike [10].

There have been some significant approaches towards the generation of synthetic figures in virtual environments, such as the work of Kalra et al.[11], which describes an interactive system for building realistic virtual humans for real time applications, and that of Aubel et al. [1], which presents techniques for rendering and animating a multitude of virtual humans in real-time.

One important application that utilises many aspects of human motion and simulation is a commercial system called Jack, developed at the University of

Pennsylvania [3]. It contains kinematic and dynamic models of humans based on biomechanical data and displays several built-in behaviours including balance, reaching and grasping, walking and running. A similar environment is HUMANOID [7], which is additionally using metaballs for a more realistic representation of the skin and muscles. VLNET [19, 18] is a networked multi-user virtual environment that is using HUMANOID's articulated body model as a basis for virtual human display and motion.

A synthetic human should not only be capable of animating its body and applying forces on other objects, but also be able to interact with a dynamic environment and exhibit some form of behaviour. Borrowing the theory from the field of Intelligent Agents [30], a Virtual Agent should be able to sense the environment and decide on its actions according to its predefined goals. One important decision technique for dynamic environments is planning [15, 17, 27] and there are a number of different systems that use planning for the behavioural control of virtual agents.

N. Badler and B. Webber propose an architecture [4] that is providing high level control of a Virtual Human with planners and parallel state-machines, and low level one with Sense-Control-Act loops. On the other hand, A. Caicedo and D. Thalmann present a complex behavioural engine for autonomous virtual agents with trust models and beliefs about other agents [9].

One interesting application towards a 3D environment with intelligent virtual agents are the Virtual Teletubbies [2]. The system uses a simple physics model to simulate gravity and the agents' personality is based on a novel behavioural architecture, the Behavioural Synthesis Architecture [6]. Agents with personality are also the main subject of the work of D. Silva et al [26]. They propose a Synthetic Actor model that connects emotions and social attitudes to personality, providing a long-term coherent behaviour. They present two games as case studies to their proposed architecture.

The EXCALIBUR project [16] uses a generic architecture for autonomously operating agents that can act in a complex computer-game environment. The agents use planning in a constraint programming framework and the behavioural model is able to handle incomplete knowledge and information gathering. Another interesting system for the creation of real-time behaviour-based animated actors is Improv [21]. It consists of an Animation Engine that uses procedural techniques to generate layered, continuous motions and transitions between them, and a Behaviour Engine that is based on rules governing how actors communicate and make decisions. The combined system provides an integrated set of tools for authoring the 'minds' and 'bodies' of interactive actors.

Lokutor [14] is an agent prototype with an embedded behaviour system for use in 3D virtual environments. It takes advantage of speech synthesis and recognition techniques to interact with the user and uses Web-based technology, such as Java3D, VRML and H-Anim [24] for portability.

There may be a considerable amount and variety of research going on in the field of intelligent virtual agents, but not all the aspects of the problem have yet been explored. Applications like Jack mainly focus on the accuracy of the simulation and include far too many details on human body structure, because their main target is the field of industrial design (and similar fields that need detailed results), while others emphasise too much on personality and behaviour. On the other hand, applications such as Virtual Environments and simple Simulation Systems do not need 100% accuracy. They require natural looking motion, as well as acceptable execution speed

and this is the aspect that our approach is focusing on. The aim of our system is to combine simple yet effective design algorithms and implementation techniques for the creation of synthetic humans with planning capabilities that focus on real-time desktop VR applications.

### 3 A Simulated Human for Real-Time Environments

The SimHuman system is a platform that allows the user to define three-dimensional scenes with an arbitrary number of virtual agents and user-controlled avatars, and serves as a basis for applications such as virtual environments and simulation systems. In this section we will focus on the design issues and functionality of the virtual agents that are implemented in the SimHuman environment. We will present issues such as motion, collision detection, inverse kinematics and planning.

#### 3.1 Display and Primitive Motion

In the SimHuman environment all objects of the scene (including virtual agents' bodies) are defined in terms of 3D Polygons. Agents are not based on a fixed model, but the system is able to load models dynamically, from a *geometry file* that contains the details of their appearance. Additionally, some skeletal information is necessary, to define how the body segments are going to be transformed. This is loaded from a supplementary file, called *hierarchy file* which includes the joint hierarchy and the position and limits of each joint. Virtual agents can have an arbitrary number of joints and segments and any possible hierarchy tree to connect them. This gives the user / programmer the ability to load various models and adjust the program to the needs of different applications having the required level of detail.

A classic problem of articulated figures is that the rotation of a segment causes a crack around the joint, because the faces of adjacent segments are not adjoining anymore. One way to deal with it is to have fixed primitives (usually spheres) on the joints, but one drawback is the fact that these primitives do not always fit perfectly with the model's meshes, thus distorting the appearance of the model. Another problem is that they add a lot more polygons to the scene and decrease the performance significantly.

In our system we have the ends of adjacent segments always connected with each other and, whenever a segment is rotated, the vertices that are common with any adjacent segment are not transformed. The effect of this method is demonstrated in figure1.

The SimHuman system is using keyframing as the basis of all animation sequences. This generates a smooth transition between two states (poses). With this simple process of keyframing, we can load various body postures stored in an *animation library* and use them to produce more complex animation sequences, such as walking.

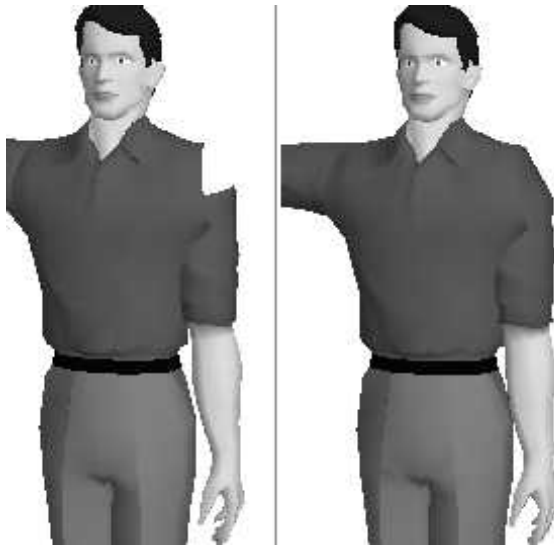


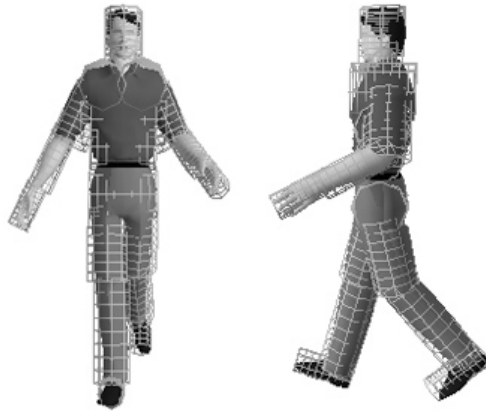
Fig. 1. Shoulder rotation without and with common vertices

### 3.2 Physically-Based Modelling, Collision Detection, and Response

In our system there is a physically based modelling engine that is used to increase the believability of the animation. The physical simulation is conducted in discrete timesteps. Each object has its own mass, position and velocity and in each timestep its position and velocity are recalculated following the laws of kinematics and collision.

In a simulated environment all objects have to behave as rigid bodies, i.e. no object should be allowed to penetrate another. The process of *collision detection* is to examine if the whole or part of the geometry of an object is within another. Associated with this is another equally important process, that of *collision response*, which should determine the new position and velocity of the two (or more) objects that collided.

Calculating an accurate collision detection and response for a human body mesh is not an easy task. The program has to check each polygon of the mesh against all the other objects of the scene and determine if it is penetrating another polygon. The approach that we use in our system is somehow different. We use bounding primitives around the human body segments and perform all collision detection checks with them. This reduces the computational time significantly, making it possible to run in acceptable speeds in real-time systems. The primitives that we have chosen are cylinders, mainly because their geometry is more symmetrical compared to bounding boxes and they still fit well around the human body (figure 2).



**Fig. 2.** A human body mesh covered with bounding cylinders

### 3.3 Dynamic Actions

Dynamic actions are those actions executed by a virtual agent that are not predefined (such as an animation sequence), but adapted to an ever-changing environment. This is, therefore, a very important issue for simulation systems, because, in most of the cases, their objects / creatures have to act in a dynamic environment. In our system, the physically based modelling engine as well as the co-existence of more than one agents and avatars inevitably ‘generate’ a dynamic environment, and a virtual agent has to be able to execute dynamic actions to interact with it.

Such actions are path planning and obstacle avoidance, i.e. the ability to move to a specified target without bumping into other objects. To execute these actions, the virtual agent has a ‘sensing’ mechanism, which enables it to have a limited view of the environment. This mechanism is based on *ray casting*. Whenever the agent ‘looks’, it casts a number of rays into the scene, and reads the objects that these rays intersect with. With this mechanism the agent is able to know the position, size and type of the objects that are in its field of view, and use this information to avoid obstacles and plan its path to the target. The agent keeps its own, internal map of the scene, which is the agent’s ‘memory’ and is updated after every ‘look’ process. The agent uses this map to calculate a safe path to the target.

More complicated dynamic actions involve catching a moving object, hitting it, or leaving an object on a specified location. These actions must use a form of inverse kinematics, because they involve more than one joints that have to be coordinated to succeed. Instead of using a generic inverse kinematics solver, a different approach is introduced. This approach tests at every step the best rotation for each joint to achieve the target.

The set of joints (or the joint chain) that is going to be used by the system is first defined. Then, a function has to be assigned and return on how close to the target is the current state of the virtual human. The process (in pseudocode) is:

```

For each joint in the chain
  For each degree of freedom of the joint
    increase and decrease angle by a value v
    check which of the two states improves the
      function
    if that state is different from the
      previous one
      decrease the value of v
    else increase it
    assign the value to the angle

```

This process is repeated in each frame and the system always corrects itself towards the target. The speed of rotation change per joint depends on the success of the move. If one segment is ‘shaking’ (the angle is increased in one step and decreased in the next, or vice versa), the joint speed is decreased to provide finer approach to the target. On the other hand, if an angle change is always heading towards the correct direction, the speed is increased until it reaches the maximum value.

### 3.4 Behavioural Control

What we have presented so far is a human model, which can execute various actions in a 3D environment. These features may be enough for avatars, but a virtual agent should also have a mechanism to select actions and execute them in order to achieve its goals. This is the agent’s ‘mind’, which defines its behaviour, and in the SimHuman system it is implemented with the use of a *planner*. Therefore, we can say that the agent consists of two parts, the motion controller and the behavioural controller.

The agent’s behavioural control unit works with a symbolic representation of the world, just like a human’s mind does not think in terms of 3D coordinates and geometry, but in rather abstract terms. Procedures such as walking towards a target, avoiding obstacles and grasping objects are not calculated at the high-level declarative part of the ‘mind’ but are part of the agent’s functionality as standard actions. Otherwise, the planning time needed to execute those actions would slow down significantly the system’s performance. A real-time system with good performance should use planning only at a higher level, while the real time activities should be implemented with standard algorithms (such as path finding, collision avoidance, inverse kinematics, etc). Nevertheless, this approach poses a very important problem: how to represent the world in abstract terms and how to maintain its integrity.

Just like an intelligent agent might be based on a sense – decide – act loop, our virtual agents have equivalent methods for sensing, deciding and acting. Nevertheless, these three processes are not executed sequentially, but in parallel. One action might take a significant amount of time to be executed (e.g. walk to a very distant target) and the `decide()` process might also take some time, because planning is not a simple task. Therefore it is much more efficient to let them work in parallel, e.g. while the agent is walking it can also think of its next actions. The `sense()` process is executed very frequently, because the agent is always sensing (looking) the place. Based on the sensed data, the `decide()` process is trying to compute a plan for the

agent's actions using symbolic information of the world. This plan is a sequence of actions executed by the `act()` process. This coordination between sensing, deciding and acting becomes more complex in dynamic worlds where the state of the world can change arbitrarily (due to other agents' or avatars' actions). In such cases the `decide()` process should also check whether the plan it has generated is still applicable, or if a change in the world's state prohibits the agent from reaching its goals. In the second case a new plan has to be computed and the action queue is not valid.

When the agent senses, it 'scans' the 3D world and 'reads' the objects. There is a list, which holds all the objects that the agent has 'seen' during its last sensing process (the agent's memory), namely the name and type of each object together with its attributes (position, orientation, size and other object-specific attributes). Nevertheless, the information stored in that list is of no use to the decision process, because the planner works with an abstract representation of the world and its objects. For example, it would need information such as 'the ball is on the table' instead of 'the ball's position is (5.3, 3.2, 3.1) and the table's position is (4.1, 1.0, 4.0)'. Therefore, there is a module that generates the abstract definition of the world and the spatial relationships between the world's entities.

As stated before, the `decide()` process uses planning to generate an action sequence that should achieve the agent's goals. First of all, the agent should have a clearly defined *goal state*, a set of objects' attributes and relationships that the agent aims for. Additionally, the planning process also needs the set of all the possible actions that the virtual agent is able to perform, and each of these actions should also state its preconditions and its effects in abstract terms. For example an action such as 'put object A on object B' has the precondition that the agent is holding object A and the effect that object A is on object B.

The sensing process stores a list of objects and their attributes, and can therefore generate an abstract world representation with all possible relationships between the objects. This set of terms that represent the world in an abstract way are the agent's *beliefs*, on which the planner is based to generate an action sequence that will lead the agent to its goals. Nevertheless, it might be possible that the planner has worked out a solution, but, during the execution of an action, the state of the world has changed in such a way that the plan is not going to succeed. This can simply be determined by comparing the set of beliefs to the actual state of the world as it is perceived by the sensing process. If the 'expected' state does not match the actual one, the plan has to be recalculated. Of course, after any successful execution of an action by the agent, this 'expected' state has to be updated as well (by applying the effects of the action to the agent's beliefs). If a new plan has been generated, all remaining actions in the action sequence are deleted and the agent starts over with a new set of orders. The `decide()` function expressed in pseudocode is:

```
bool decide() {
    if(beliefs_match_sensing_data) {
        if(goals_match_beliefs) {
            // the plan is applicable and has succeeded
            clear(action_queue);
            return true;
        }
        // else: the plan is still applicable, but has
```

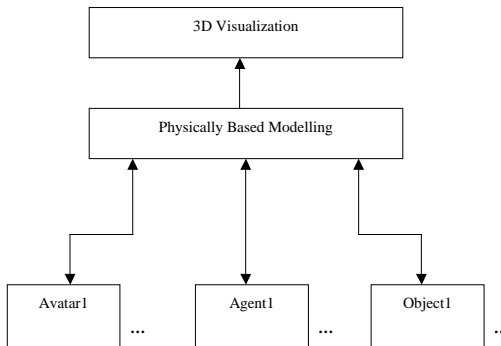
```

not
    succeeded yet
    return false;
}
// the plan is not applicable, it has to be
recomputed
clear(action_queue);
update(beliefs);
recompute(plan);
update(action_queue);
}

```

## 4 The Architecture of SimHuman

SimHuman consists of two basic modules: the 3D Visualization, which renders the scene, and the physically based modelling, which applies the laws of physics. Each scene can have an arbitrary number of passive objects, virtual agents and avatars (figure 3). During each timeframe, the physically based modelling part applies forces, checks for collisions between objects and handles their motion. The 3D Visualization part then uses the final positions of the world's entities and displays the scene on the screen.



**Fig. 3.** The general architecture of SimHuman

Both virtual agents and avatars are articulated figures (humans or other lifelike characters) and use therefore a geometry file, a joint hierarchy file and an animation library to execute their actions. All the other objects of the scene are described by simple geometry files. A virtual agent consists of a motion controller, which is able to execute simple and dynamic actions, and a behavioural controller, which decides for the agent's behaviour in the virtual space. On the other hand, an avatar has a motion controller with the same structure, but its behaviour is determined directly from the user. Figure 4 shows the architecture of an agent and an avatar. Boxes stand for modules and drums for data

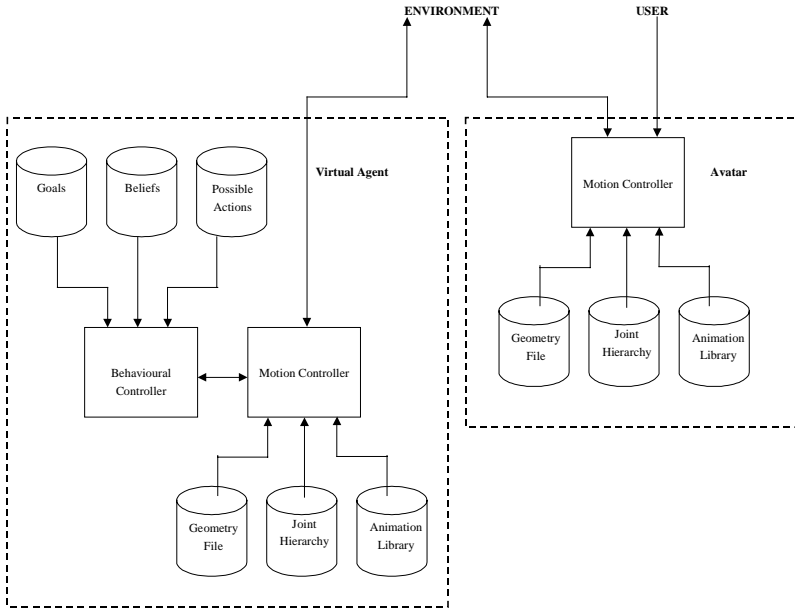


Fig. 4. The architecture of virtual agents and avatars

### 5 An Illustrative Example

In this section we will present a sample application that has been created with SimHuman. It is the typical planning problem of blocks world: there is a set of boxes on a table and the agent’s task is to move the boxes and achieve a specific arrangement. One example is that of figure 5.

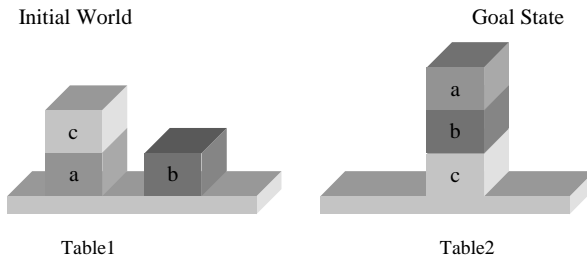


Fig. 5. A blocks world example

In this example, one symbolic representation of the initial world is { on (a,table1), on(c,a), on(b,table1), clear(c), clear(b) } and of the goal state { on(c,table2), on(b, c), on(a, b) }, where

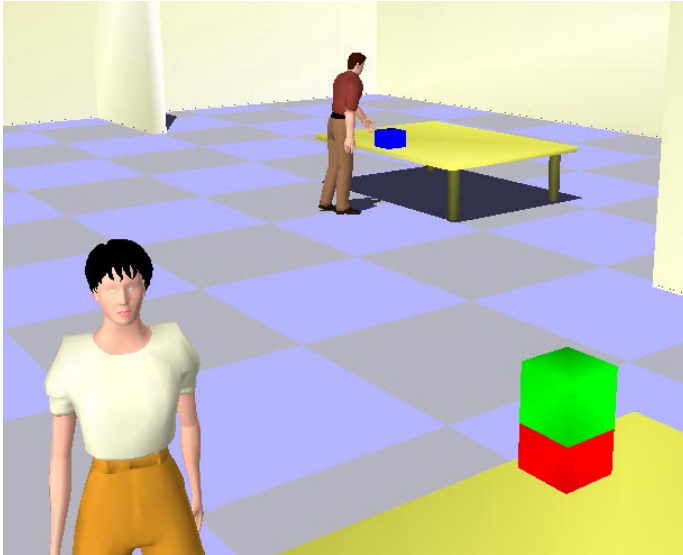
$\text{on}(A,B)$  means that object A is on object B and  $\text{clear}(X)$  means that there is nothing on top of object X. The possible actions for an agent are:

- to pick up a block that is clear ( $\text{pickup}(\text{Obj})$ )
- to put a block on another clear block or the table ( $\text{put}(\text{Obj}, \text{Location})$ )

Therefore, a possible plan would be  $\{ \text{pickup}(c), \text{put}(c, \text{table2}), \text{pickup}(b), \text{put}(b, c), \text{pickup}(a), \text{put}(a, b) \}$ .

We use a scene with two tables and three boxes, where the agent's task is to achieve a specific arrangement. When the program is running, the agent is receiving information about the scene's objects through sensing. It, then, converts this information into symbolic relations (e.g. it generates all the  $\text{on}(X,Y)$  relations by comparing the objects' positions and bounding boxes) and lets the planner find a solution. This solution is then converted to specific agent actions (e.g. in order to pick up an object, the agent has to go to a place near the object and grasp it).

An extension of this example is to have two agents with contradicting goals. In this case, agents will always have to recompute their plans and follow a new set of actions (figure 6).



**Fig. 6.** A screenshot of the blocks world 3D environment

## 6 Implementation

We have implemented the SimHuman system as a set of C++ classes, using OpenGL for the 3D visualization. We have created a semi-automated platform in Microsoft Visual C++, which demonstrates the capabilities of the SimHuman system in a simulated physical environment and allows the development of 3D environments with virtual agents [28]. The geometry models for virtual humans can be directly imported from Curious Labs Poser [22], provided that they have been exported as VRML 97

files, due to an embedded VRML parser that is used by the program. Therefore there is practically no limitation on the number of different models that can be used by our system. The SimHuman platform automatically generates the bounding cylinders of the body and, together with a hierarchy file and an animation library, uses the 3D model as a representation of an agent or avatar.

Our planner is implemented in Prolog. The planner itself as well as the abstract action definition are precompiled in SICStus Prolog [25], while the current world state and goals are dynamically passed as arguments whenever the agent has to calculate a new plan. The ability to call SICStus Prolog queries from a C program allows SimHuman to use the planner dynamically while running.

## 7 Conclusions and Future Work

In this paper we have presented a design and implementation approach for virtual agents, which can be used in real-time systems. The field of human modelling, simulation and behaviour is an area of continuous research and a lot of different approaches have been proposed. In the SimHuman system, we have tried to balance between efficiency and accuracy in order to produce believable human motion in real-time environments. Possible applications include simulated worlds that demonstrate human action, virtual environments with avatars or other computer controlled human models, educational applications, etc.

There are, nevertheless, a lot of issues that need to be solved, such as handling of uncertainty, execution of parallel actions, hierarchical planning, etc. We are working on integrating spatio-temporal reasoning and planning techniques into SimHuman's behavioural control and on allowing concurrent execution of agents' actions. In order to do this, we intend to follow the paradigm of Advisor [13] and TRL-tutor [20], temporal planners developed by members of our research team. Additionally, we are working towards a generic visual tool for the set up of a scene and the description of agents' behaviour to serve as a basis for the automatic generation of virtual environments with believable agents.

**Acknowledgements.** This work has been supported by the Greek Secretariat of Research and Technology under the PENED'99 project entitled "Executable Intensional Languages and Intelligent Multimedia, Hypermedia and Virtual Reality applications", Contract No. 99ED265.

## References

1. Aubel, A., Boulic, R., Thalmann, D.: Real-timeDisplay of Virtual Humans: Level of Details and Impostors. *IEEE Trans.Circuits and Systems for Video Technology*, Special Issue on 3D Video Technology
2. Aylett, R., Horrobin, A., O'Hare, J., Osman, A., Polshaw, M.: Virtual Telebubbies: reapplying a robot architecture to virtual agents. In *Proceedings of the Third International Conference on Autonomous Agents*. New York: ACM Press (1999) 514-515
3. Badler, N., Phillips, C., Webber, B.: *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press (1993)

4. Badler, N., Webber, B., Becket, W., Geib, C., Moore, M., Pelachaud, C., Reich B., Stone, M.: Planning and parallel transition networks: Animation's new frontiers. In: S. Y. Shin and T. L. Kunii (eds.), *Computer Graphics and Applications: Proc. Pacific Graphics '95*, World Scientific Publishing, River Edge, NJ. (1995) 101-117
5. Badler, N.: Virtual humans for animation, ergonomics, and simulation. *IEEE Workshop on Non-Rigid and Articulated Motion*. Puerto Rico (1997)
6. Barnes, P.: A behaviour synthesis architecture for cooperant mobile robots. *Advanced Robotics and Intelligent Machines*. J.O. Gray, D.G. Caldwell (eds). *IEE Control Engineering Series 51* (1996) 295-314
7. Boulic, R., Huang, Z., Shen, J., Molet, T., Capin, T., Lintermann, B., Saar, K., Thalmann, D., Magnetat-Thalmann, N., Schmitt, A., Mocozet, L., Kalra, P., Pandzic, I.: A system for the parallel integrated motion of multiple deformable human characters with collision detection. *Computer Graphics Forum* 13(3). (1995) 337-348
8. Brogan, D., Metoyer, R., Hodgins, J.: Dynamically Simulated Characters in Virtual Environments. *IEEE Computer Graphics and Applications*. September/October 1998, Volume 15 Number 5 (1998) 58-69
9. Caicedo, A., Thalmann, D.: *Virtual Humanoids: Let Them be Autonomous without Losing Control*, Proc. 31A2000, Limoges, France (2000)
10. Hodgins, J., Wooten, L.: *Animating Human Athletes*. In *Robotics Research: The Eighth International Symposium*. Y. Shirai, S. Hirose (eds). Springer-Verlag: Berlin (1998) 356-367
11. Karla, P., Magnetat-Thalmann, N., Mocozet, L., Sannier, G., Aubel, A., Thalmann, D.: Real-time Animation of Realistic Virtual Humans. *IEEE Computer Graphics and Applications*, Vol.18, No.5. (1998) 42-55
12. Magnetat-Thalmann, N., Carion, S., Courchesne, M., Volino, P., Wu, Y.: Virtual Clothes, Hair and Skin for Beautiful Top Models, *Computer Graphics International '96*, Pohang, Korea (1996) 132-141.
13. Marinagi, C.C., Panayiotopoulos, T., Vouros, G.A., Spyropoulos, C.D.: *Advisor : A knowledge-based planning system*. *International Journal of Expert Systems, Research and Applications*, Vol.9, No.3. (1996) 319-355
14. Milde, J.: *Lokutor: Towards a Believable Communicative Agent*, *AGENTS 2000 Workshop*, Barcelona (2000)
15. Nareyek, A.: A Planning Model for Agents in Dynamic and Uncertain Real-Time Environments. *Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments at the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98)*, Technical Report, WS-98-02, 7-14. AAAI Press, Menlo Park, California (1998)
16. Nareyek, A.: *Intelligent Agents for Computer Games*. In *Proceedings of the Second International Conference on Computers and Games* (2000)
17. Panayiotopoulos, T., Katsirelos, G., Vosinakis, S., Kousidou, S.: An Intelligent Agent Framework in VRML worlds. *Advances in Intelligent Systems : Concepts, Tools and Applications*, S. Tzafestas (ed.), Kluwer Academic Publishers (1999) 33-43
18. Pandzic, I. , Capin, T., Magnetat Thalmann, N., Thalmann, D.: Motor functions in the VLNET Body-Centered Networked Virtual Environment, *Proc. 3rd Eurographics workshop on Virtual Environments, Monte Carlo, Virtual Environments and Scientific Visualization '96*, Springer, Wien. (1996) 94-103
19. Pandzic, I., Capin, T., Magnetat Thalmann, N., Thalmann, D.: VLNET:A Networked Multimedia 3D Environment with Virtual Humans. *Proc. Multi-MediaModeling MMM '95*, Singapore. (1995) 21-32
20. Payiotopoulos, T., Avradinis, N., Marinagi, C.C.: Using Forward Temporal Planning for the production of Interactive Tutoring Dialogues. *Advances in Intelligent Systems : Concepts, Tools and Applications*, (S. Tzafestas ed.), Chapter 20, Kluwer Academic Publishers, Netherlands. (1999) 219-230.

21. Perlin, K., Goldberg, A.: Improv: A system for scripting interactive actors in virtual worlds. In ACM Computer Graphics Annual Conf. (1996) 205-216
22. Poser 4: <http://www.curiouslabs.com/products/poser4>, Curious Labs
23. Pourazar, G.: A method to capture and animate the dynamics of human motion. *COMPUGRAPHICS* 91, I (1991) 181-197
24. Roehl, B.: Specification for a standard humanoid [Online]. Available: <http://ece.uwaterloo.ca/~h-anim/spec1.1/>
25. SICstus Prolog: <http://www.sics.se/sicstus>
26. Silva, D., Siebra, C., Valadares, J., Almeida, A., Frery, A., Ramalho, G.: Personality-Centered Agents for Virtual Computer Games, *Virtual Agents 99*, Workshop on Intelligent Virtual Agents, Salford, UK (1999)
27. Vosinakis, S., Anastassakis, G., Panayiotopoulos, T.: DIVA: Distributed Intelligent Virtual Agents, *Workshop on Intelligent Virtual Agents, Virtual Agents 99 Salford* (1999) 131-134
28. Vosinakis, S., Panayiotopoulos, T.: Design and Implementation of Synthetic Humans for Virtual Environments and Simulation Systems. 5th WSES/IEEE WORLD MULTICONFERENCE ON Circuits, Systems, Communications & Computers (CSCC 2001), to be presented (2001)
29. Welman, C.: Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation. MSc Thesis, Simon Fraser University (1993)
30. Wooldridge, M., Muller, J., Tambe, M., editors: *Intelligent Agents II, Agent Theories, Architectures and Languages*. Volume 1037 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag (1996)