Tutorial Book of Virtual Concept 2006
Cancún, Mexico, November 30th – December 1st, 2006

Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation

Fernando S. Osório [(1)], Soraia R. Musse [(1)], Renata Vieira [(1)], Milton R. Heinen [(1)], Daniel C. de Paiva[(2)]

(1) : UNISINOS – Applied Computing - PIPCA
Av. Unisinos, 950 – Bloco 6B – São Leopoldo, RS
BRAZIL
+55 (51) 3591.1100 / Fax: +55 (51) 3590.8162
E-mail : {fosorio, soraiarm,renatav}@unisinos.br
miheinen@gmail.com

(2) : FAGOC Campus Ubá e UNIPAC
PhD Student at Poli-USP – PSI
São Paulo - BRAZIL
+ 55 (32) 3531.2370 / Fax: +55 (32) 3551.1111
E-mail : nartos@gmail.com

Abstract

This tutorial aims to present new trends, methods and applications related with the interaction of agents and objects present in a Virtual Reality (VR) Environment. In the first part of this tutorial we will discuss about the introduction of interaction based on physics, including concepts related to perception, action, kinematics and dynamics (including rigid body dynamics, flexible/deformable objects and particles systems). After this discussion about physical interaction between VR agents and elements, then the second part of the tutorial will focus on the behavioural simulation of virtual autonomous agents. We will discuss about different simulation techniques of agent behaviour control, including autonomous agent control architectures (e.g. deliberative, reactive and hybrid architectures). The introduction of knowledge about the agents (e.g. emotional states, personality, personal profile) and about the environment (e.g. special places, functioning rules, place profile), will be also addressed. The knowledge introduced is then used to improve aspects related to the agents' autonomy, the interaction within the VR environment, and the degree of reality in the VR simulations. We conclude this tutorial with some examples of practical applications, including recently developed VR applications implemented by our research group.

Table of Contents

Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation

I. Introduction

This tutorial aims to present and to demonstrate the importance of physical and behavioural simulation techniques integrated into Virtual Reality (VR) environments, in order to increase the realism obtained in these virtual environments. A Virtual Reality environment should provide to users tools for visualization, interaction and immersion in a 3D environment, but it should also be able to reproduce the real world as realistic as possible. So, to make possible a reliable reproduction of the real world in a virtual environment, it is necessary to introduce in this virtual environment some elements and properties from our world, in other words, the virtual elements should behave and react in a very similar way related to the physical elements of the real world. Then, our main concern here is to emphasize how important is to integrate physical simulation models into virtual environments, as well as, behavioural simulation models of autonomous agents. Virtual agents' behavioural simulation can be obtained based on the integration of Artificial Intelligence techniques, applied to these virtual environment elements. The behavioural models allow us to create virtual environments with human characters and autonomous agents, endowed with the capacity to perceive, to act and to interact with other VR environment elements, as well as, to interact with the users of the system.

Several authors support a similar proposition as described above [Funge 1999, Watt 2001], which should be adopted in all types of VR applications: from VR tools used to help in the design of new products or to simulate real world situations, until VR applications in the area of digital entertainment and games. The Figure I-1 presents the hierarchy of models that should be used to create virtual worlds, according to John Funge's proposal [Funge 1999**]**. Funge proposes an hierarchy of levels, composed by the geometry, kinematic, physical, behavioural and cognitive level. This hierarchy demonstrates the whole complexity of realistic virtual environments implementations, since it is very important to have implemented the base levels of this pyramid in order to support and achieve more complex models (more realistic), that are in the top of the pyramid. We found also a similar scheme described by A. Watt and F. Policarpo, where they define intelligent behaviours and hierarchies in behaviour control [Watt 2001, pg. 486]. In this tutorial we adopted this concept of hierarchy of levels applied to create realistic models of simulation in virtual worlds, however we will go deeper and detail each one of these levels (geometric, physical, behavioural and cognitive). So, we seek to

obtain a better proposal to describe and structure the design and practical development of virtual environments, and also its application into virtual simulations tasks. Besides that, we will present also some practical applications, developed in our research group, which emphasize the importance of the physics, behavioural and cognitive models simulation, in order to implement realistic simulations and VR product design.

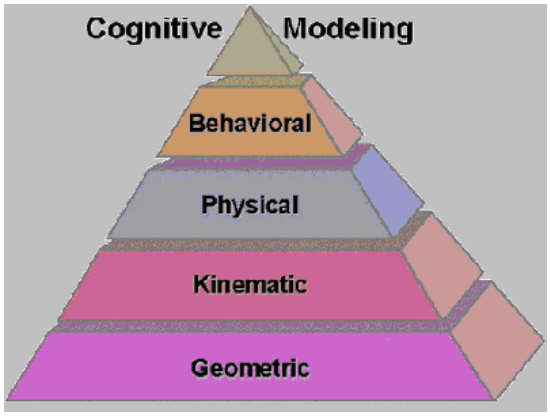

Figure I-1: Hierarchy of models used to create virtual worlds [Funge 1999]

This text is organized as follows. In the next section (II) we describe the main concepts and components related to this tutorial: simulation models that include geometry, physics, behaviour, knowledge and cognition in order to improve the realism in VR environments. Section III describes some specific tools and techniques used to implement physics simulation in virtual environments. In addition, we present in the next section (IV) the concepts related to the techniques applied in order to control the virtual agents (intelligent objects and characters), obtaining different types of intelligent behaviours and agents' autonomy. Section V describes the introduction of high-level knowledge in virtual environments, obtained through the use of ontology-based VR simulations. In the next section (VI), named practical applications, some of our research group VR tools practical implementations are described, including examples of VR based simulation of physics, behaviours, knowledge and cognition. The applications are divided into two main groups, (*i*) autonomous robots design based on VR simulation, and (*ii*) application of ontology into a VR environment, simulating a dense populated environment. We conclude this work with some final remarks and discussions in the section VII.

"***We live in a material world***"
which is governed by physical laws…
which is populated by different types of creatures…
which have different types of behaviours and intelligence.
This is our real world.

II. Virtual Reality and Realistic Simulations

In this section we will describe the importance to add properties to the virtual environment that allow us to integrate more advanced characteristics from our real world. We move from the concept of a 3D visualization tool (with human sensorial immersion in a virtual world) to the concept of 3D simulation tool (with human immersion in a "physical and realistic" virtual world). The main objective of this models integration (physical and behavioural), simulating elements from the real world, is to increase the realism of the user immersion experience in a virtual environment. This can approximate the VR applications to our concrete world. The integration of these models into VR applications is obtained through the implementation of physical and behavioral computational simulated models into the virtual environment.

II-1. Geometry

Simple VR environments, denominated here as "**VR visualization**", just look for to provide a visualization of the virtual world, with the maximum visual realism possible, in a way to create an immersion sensation in this world. These environments can be classified as advanced visualization tools and they are focused mainly in the computer graphics information processing: the key-elements are object geometries which are represented by polygons and 3D meshes. **VR visualization** environments can enclose virtual worlds composed only of static objects, where the virtual environment elements don't possess a movement or behavior. In some of these environments we can also find animated objects, usually based on "pure" classic techniques of computer graphics animation [Foley 1990, 1993; Watt 1993], in other words, all the movements are previously established, describing paths and behaviors at the geometric level (e.g. key-frame animation). Therefore, in this type of virtual environments, the virtual world is considered as a simple three-dimensional description of the real world, without any abstract concept used to represent the virtual elements: objects are simple 3D meshes and don't have any main function, meaning/semantic or autonomy associated to them.

II-2. Physics

A second abstraction level can be added to virtual environments, reaching what we called here the "**VR physical**" (simulation of motion based on physics). When we begin to consider the world composed by concrete objects (virtual elements are not just polygons in the 3D space), and we begin *to associate behaviors to these objects, according to the physics laws*, then we will have a "**VR physical**" environment. Our physical (concrete) world is formed by objects that possess physical properties, such as:

- Physical structure, physical resistance, mass, density, elasticity, etc;
- Position and orientation in the 3D space;
- Linear and angular velocities;

- Movement and acceleration/deceleration with application of forces and torques;
- Kinetic and potential energy;
- Attraction and repulsion (e.g. gravity, magnetic charges);
- Inertia, friction and collision;
- Laws of conservation of energy, momentum, angular momentum and mass.

Virtual Reality systems should simulate the movement of rigid bodies, apply forces and simulate object kinematics, detect and react to collisions (since two concrete objects cannot occupy the same space at the same time and they must respect the laws of conservation of energy), simulate articulations and object deformations (changing the object shape or even breaking it into smaller pieces). The implementation of all those physical behaviours must be done in order to increase the virtual worlds realism [Bourg 2002, Watt 2001]. In this way, the objects and elements of the virtual world start to behave as the concrete elements of our real world (e.g. a virtual car should behave like a real car). When applying physics simulation, the virtual objects start to interact with each other (automatically), according to their properties, and the user can also start to interact, in a more natural way, with these objects present in the virtual environment.

II-3. Behaviour

The third abstraction level is related to the agents' behavior (intelligent objects and autonomous characters) that are present in the virtual environment. That's what we called "**VR Behavioural**" environment. The agents should exhibit at least a simple level of intelligence, based on behavioral control techniques derived from the Artificial Intelligence (e.g. A*, FSA, BDI, scripts, steering behaviors, intelligent objects, reactive, deliberative and modular-hybrid control architectures) [Osorio 2005, Garcia 2003, DeLoura 2000, Thalmann 1999, Reynolds 1999].

Different types of agents can exhibit different types of behaviors: if we consider the agents as being virtual people, they can exhibit a series of intelligent behaviors, some simpler and other more sophisticated; if we consider other types of creatures (animals and other live creatures in general) they can behave in a different way; or even industrialized objects with their own functionalities and operational way, could also behave autonomously in a specific way. All these agents could have several different types of behaviors, from the most elementary to the more complex.

A common property in the **VR Behavioural** environments is that the elements of the virtual environment stop being only simple objects subject to external forces imposed by the user, or considering a set of predefined static forces. The virtual agents start to perceive the environment, to decide and plan their actions, and to execute them. The agents *incorporate a sensorial capacity and a motor control*, achieving a more sophisticated behaviour that possesses a certain degree of autonomy [Russell 1995, Wooldridge 1995]. Autonomous agents can control their own movements and behaviours.

II-4. Cognition and Knowledge

The fourth and last abstraction level is the one called **"VR Cognitive"**. In this level, the agents start to have high level "reasoning" capacities, in terms of Artificial Intelligence. They could *use sophisticated methods to represent knowledge* about the world where they are inserted, and also they could *use sophisticated methods to manipulate this knowledge*. The VR environments, at this level, start to make use of knowledge representation models (e.g. ontology, informed environments) [Paiva 2005, Farenc 1999], to describe the agents (e.g. emotional states, personality, personal profile) and the environment (e.g. special places, functioning rules, place profile). For example, we can add information about the environment describing the opening hours of a store, or the security measures associated to some flammable liquids (e.g. do not smoke nearby these objects, or provoke sparks).

Agents in **VR Cognitive** environments can deal with different reasoning mechanisms (e.g. inference, analogy), have long-term and short-term memory, communicate with other agents or users, and even they can have the capacity to adapt and to improve themselves (e.g. learning, evolution) [Osorio 2005; Heinen 2006a, 2006d]. One of the most valued attributes in this type of environments is that it is possible "to populate" the virtual environment with several autonomous agents, creating multi-agents systems [Weiss 1999] and crowd systems [Musse 2001]. The presence of multiple agents in a virtual environment, that respect an adequate social behaviour, is an interesting way to increase realism in VR environments. Note that each agent, besides their elementary behaviors, could also have mechanisms to communicate, to cooperate and to coordinate their individual and collective (group) behaviours.

The Figure II-1, presented in the next page, shows a complete scheme structuring the features, attributes and functionalities of each one of those four levels of abstraction above described in this section. In the next sections, some tools, concepts and application examples will be presented, demonstrating how important is to improve the realism in VR environments by adding these four levels of abstraction: VR visualization, VR physical, VR Behavioural e VR Cognitive.

In the next section (III) will be described some tools that can help and support the implementation and simulation of physical behaviours (VR physical).

From Simple VR Visualization Tools to Realistic VR Simulation Tools

Visualization	**Geometry** [3D Meshes]	Static Objects Animated Objects (Key-Frame)
Simulation of Motion	**Physics** [3D Objects]	Rigid Body (Physically based) Kinematics (Movement) Collision (Solid Objects) Collision Response Articulations Particles (Fire, Smoke, Water) Springs (Mass-spring Systems) Deformable Objects (Cloths, Elastic, Fluids) External Forces: Interaction Interaction Object x Object Interaction Camera x Object Interaction User x Object Interactive Control
Simulation of Behavior	Artificial Intelligence **"Simple A.I."** **Behavior** [Agents] [Characters]	Agents Control Scripts Finite State Automata (FSA) Perception (Sensorial) Action (Motor) Control: Reactive Control: Deliberative Control: Modular / Hybrid Memory, Beliefs, Intentions,... Biomechanics Simple Autonomous Agents
Simulation of Intelligent Behavior	Artificial Intelligence **"Advanced A.I."** **Cognitive** [Autonomous Agents] [Multi-Agents]	Knowledge Reasoning Cognition Communication Cooperation Coordination Adaptation: Learning, Optimization, Evolution Robust Autonomous Agents

Figure II-1: Models and Components of a Virtual Reality Environment applied into Realistic Simulations

III. Physics Simulation Tools

In this section we will describe some tools that can be used in order to implement and to give support to the simulations in the physical level of VR environments (real-time simulation). The main goals of these tools are to offer an "easy-to-use" framework which we can be used to simulate realistic physical behaviors in 3D virtual environments.

The physical simulation tools and frameworks usually should include models that implement [Bourg 2002, Watt 2001, Osher 2002, Fedkiw 2006]: (*i*) Kinematics Simulation: allows to apply forces over objects in order to generate motion and trajectories, acceleration, deceleration, always considering the laws of energy conservation, and also gravity, friction, collisions and reaction to collisions (see section II-2). Some tools can also provide simple steering models, which can be specific to simulate vehicles like wheeled cars, aircrafts, rockets and projectiles, boats and ships; (*ii*) Articulated Rigid Bodies Simulation, like and skeleton or an robotic arm; (*iii*) Dynamic Simulation of Deformable Objects: allows to apply forces over deformable and elastic objects, using techniques like finite-elements simulation (or fast optimized approximations of this model) and coupled mass-spring systems; (*iv*) Fluid simulation and Particle Systems: allows to simulate the interaction of complex elements, like fire, smoke, clouds and liquids, with the elements present in the environment.

There are several function libraries, APIs (Application programming interface), SDK (Software development kit) and Physics Engines used to simulate physical behaviours in real-time. These software packages are very useful in computer games development and also to implement VR applications. In this section we chose to present three widely adopted software packages: OpenSteer library, ODE (Open Dynamics Engine) and AGEIA PhysX (Integrated Hardware and Software solution). The ODE Engine was also used in our research applications described in section VI.

III-1. OpenSteer

OpenSteer[3] [Reynolds 2006, 1999; OpenSteer 2006] is a C++ library to help construct steering behaviors for autonomous characters in games, animations and virtual environments. In addition to the library, OpenSteer provides an OpenGL-based application called OpenSteerDemo which displays predefined demonstrations of steering behaviors. The user can quickly prototype, visualize, annotate and debug new steering behaviors by writing a plug-in for OpenSteerDemo.

OpenSteer provides a toolkit of steering behaviors, defined in terms of an abstract mobile agent called a "vehicle." Sample code is provided, including a simple vehicle implementation and examples of combining simple steering behaviors to produce more complex behavior. OpenSteer's classes have been designed to flexibly integrate with existing game engines by either layering or inheritance.

[3] OpenSteer - http://opensteer.sourceforge.net/

This toolkit can be used together with virtual reality environments implementations. Although the physical simulation are quite simple and very limited (Fig. III-1), this tool also provides some behavioural simulations of the next level (VR Behavioural), allowing to implement simulated boids (flocking - bird like objects), with interesting multi-agent coordinated group behaviours.

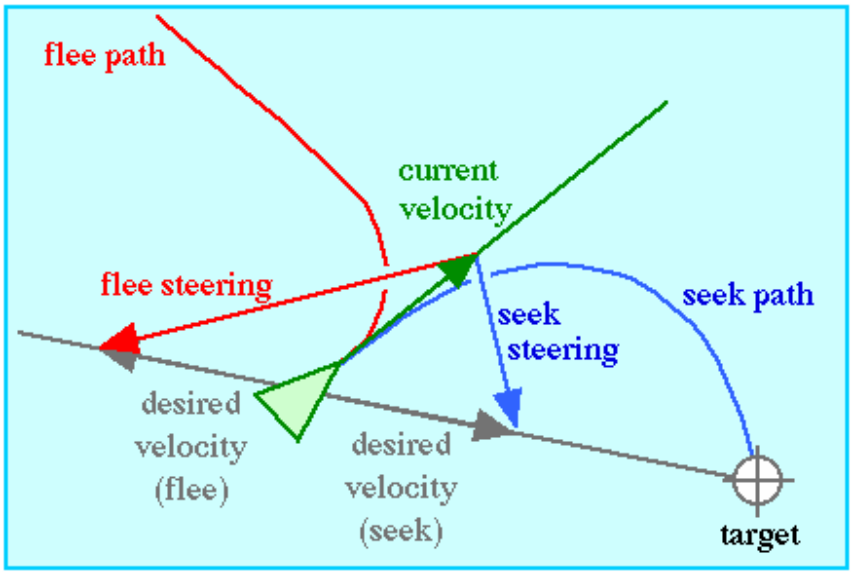

Figure III-1: Seek and Flee Behaviour in OpenSteer [Reynolds 1999]

III-2. Open Dynamics Engine (ODE)

The ODE (Open Dynamics Engine) [ODE 2006] is a free library for simulating articulated rigid body dynamics, developed by Russell Smith [Smith 2006, ODEWiki 2006]. An articulated structure is created when rigid bodies of various shapes are connected together with joints of various kinds. ODE is designed for use in interactive or real-time simulation. It is particularly good for simulating moving objects in changeable virtual reality environments, because it is fast, robust and stable. Additionally, the user has complete freedom to change the structure of the system even while the simulation is running. ODE uses a highly stable, first order integrator, so that the simulation errors should not grow out of control. The physical meaning of this is that the simulated system should not “explode” for no reason.

The simulation is based on a method where the equations of motion are derived from a Lagrange multiplier velocity based model [Wolff 2003]. ODE has hard contacts, which means that a special non-penetration constraint is used whenever two bodies collide. Another, in simulators widely used method, is virtual springs to represent contacts. ODE has a built-in collision detection system with sphere, box, capped cylinder and plane as the collision primitives. Other features of ODE are arbitrary mass distribution of the rigid bodies, and a contact/friction model based on the Dantzig LCP solver described by [Baraff 1997].

The joint types implemented in ODE are ball-and-socket, hinge, hinge-2, fixed, prismatic slider and angular motor. The hinge-2 joint is the same as two hinges connected in series, with different hinge axes. The ODE library has a native C interface (even though ODE is mostly written in C++) and platform specific optimizations.

A typical simulation will proceed like this [Wolff 2003]:

1. Create a dynamics world.
2. Create bodies in the dynamics world.
3. Set the state (position etc) of all bodies.
4. Create joints in the dynamics world.
5. Attach the joints to the bodies.
6. Set the parameters of all joints.
7. Create a collision world and collision geometry objects, as necessary.
8. Create a joint group to hold the contact joints.
9. Loop:
    a. Apply forces to the bodies as necessary.
    b. Adjust the joint parameters as necessary.
    c. Call collision detection.
    d. Create a contact joint for every collision point, and put it in the contact joint group.
    e. Take a simulation step.
    f. Remove all joints in the contact joint group.
10. Destroy the dynamics and collision worlds.

From the physic's point of view, an articulated body is simply an assembly of many rigid bodies connected together with some joints. Each one of these bodies can interact with its neighbours (collision is also avoided): a force or a torque applied on one body will also affect its connected neighbours, and all the bodies must be able to bounce into each other. Finally all the bodies have to be affected by a force of gravitation. This guarantees that when no forces are applied over the articulated body it will fall down, and, according to the forces applied into the different elements and articulations (motors), we can avoid the collapse of the articulated body, controlling its global state and position. In others words, using this tool to simulate articulated rigid body dynamics will guarantee a more realistic physical behaviour.

One can easily imagine that creating such a tool is a huge project in itself. Hopefully we can count on tools like that, since there are many fields in which rigid body simulations are required. Virtual Reality environments can embed physically based simulations of objects and forces applied to them in a very easy way: we create virtual objects duplicated in the ODE space and in the VR space. We start applying forces to the objects, passing this information to the ODE space, then ODE will integrate all the numerical data an generate new positions and orientations of each object. Using the updated information of the positions and orientations of each object we now can update the state in the VR space. The physics laws are applied in the ODE space (e.g. collision, kinematics, friction, gravity, etc) to the objects, so the visualization tools needs only to focus the attention to the graphical exhibition of the objects (e.g. textures, lights, shades, etc).

The above text described the main features included in ODE. In order to have a more complete description about this tool, we present here a more accurate definition of the two main concepts available in ODE: The bodies and the joints.

III-2-1. The Bodies

As explained in ODE's manual, a rigid body has the following properties:

- A position vector that correspond to the body's center of mass;
- A Linear velocity;
- An orientation of a body;
- An angular velocity vector, which describes how the orientation changes over time.

The previous properties are changing over time, but others are usually constant over time:

- The mass of the body;
- The position of the center of mass (relative to the body);
- An inertia matrix that describes how the body's mass is distributed around the center of mass.

These properties allow simulating objects motion in the virtual world.

III-2-2. The Joints

The rigid bodies described in the previous section can be connected together by a large variety of joints, like the ball and socket joint, the hinge joint or the universal joint. The Figure III-2 shows some of the joints available in ODE. The connection between objects is “respected” (maintained) in the simulations, so we can easily create articulated bodies, like walking creatures (e.g human characters or also animals), as well as, create complex articulated machines (e.g. cranes, bulldozers and robotic arms). The simulation of the articulated bodies is very simple, since ODE guarantees to avoid collision (interpenetration) and also can establish operational limits to the articulations.

Figure III-2: Some of the joints available in ODE

III-3. AGEIA PhysX

The AGEIA PhysX processor[4] (PPU – Physics Processing Unit) [PhysX 2006] is the critical hardware element required for optimized physics simulation and is the third engine (see Fig. III-3) of the Computer Graphics and Virtual Reality Triangle (or "Gamming Power Triangle"), which will improve realism in 3D immersive application.

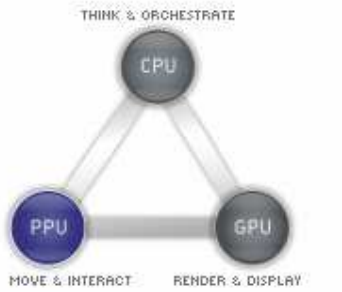

Figure III-3: Computer Graphics and Virtual Reality Triangle [AGEIA 2006]

Delivering physics in simulations is no easy task. It's an extremely compute-intensive environment based on a unique set of physics algorithms that require tremendous amounts of mathematical and logical calculations supported by massive memory bandwidth. Thus, it may be used the AGEIA PhysX processor: a specialized accelerator dedicated solely to delivering rich immersive physical simulation environments with features such as [AGEIA 2006]:

- Complex rigid body object physics system: dynamics and collision detection
- Joints and springs. Characters with complex, jointed geometries for more life-like motion and interaction
- Volumetric fluid creation and simulation
- Cloth that drapes and tears the way you would expect it to
- Smart particles. Dense smoke and fog that billow around objects in motion. Explosions that cause dust and collateral debris

A way to get real physics with the scale, sophistication, fidelity and level of interactivity is with the AGEIA PhysX processor [AGEIA 2006], which was developed to accelerate the highly specialized physically based simulations. The PhysX functionalities are accessible through the PhysX API. The Figure III-4 shows some screenshots of the AGEIA PhysX effects.

Figure III-4: Screenshots of the AGEIA PhysX effects [AGEIA 2006]

[4] AGEIA PhysX - http://www.ageia.com/physx/

III-4. Deformable Objects and Particle Systems

The modelling of natural phenomena such as water and fire for computer graphics applications remains a major challenge. The complexity of the motion exhibited by these phenomena defies the ability of animators to realistically animate by hand. The ever increasing use of computer animation in feature films to create photorealistic effects in their own right and to supplement practical elements previously filmed have motivated researchers in computer graphics (CG) to examine the extensive computational fluid dynamics (CFD) literature for algorithms which can be adapted for use in an animation environment [Enright 2002].

An important criteria for the use of such algorithms is the ability to robustly model fully three dimensional effects on the coarse computational grids commonly used in a CG environment. Early research [Fedkiw 2001, Foster 2001, Nguyen 2002,] has shown promise that when appropriate CFD algorithms are coupled with the Level Set related methods [Osher 2002], the long sought after goal of the CG community of photorealistic fire and water behavior can be attained. The Figures III-5 to III-7 shows some examples of deformable objects, particle systems and fluids simulation created at the Stanford Computer Graphics Laboratory [Fedkiw 2006].

Figure III-5: Examples of Complex Deformable Objects [Fedkiw 2006]

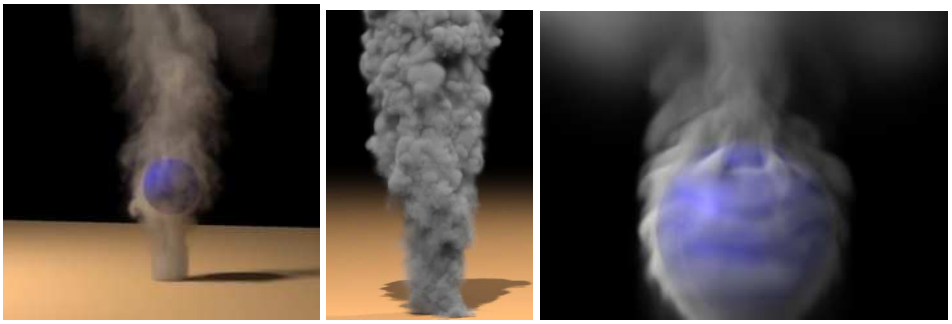

Figure III-6: Examples of Complex Particle Systems [Fedkiw 2006]

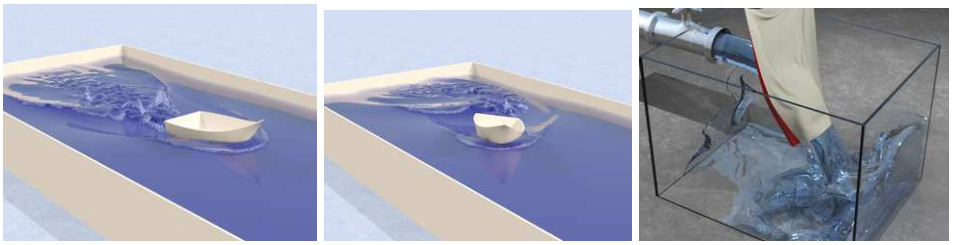

Figure III-7: Examples of Fluids Simulation [Fedkiw 2006]

The main problem with these complex dynamical simulations are that usually they are not possible to obtain in real-time or integrated into an interactive VR environment.

IV. Intelligent Behaviour

Virtual environments can be populated with several different animated elements, when these components are animated virtual characters (humans, animals or even intelligent objects) with a specific behaviour, they usually are called "virtual agents", or in games terminology, they are called NPCs (Non-Player Characters). The definition of what is a "virtual agent" or what is an "intelligent behaviour" is complex, but usually there is a consensus related to some important properties that the agents *must have*: the capacity to **perceive** the environment (input) and the capacity to **act** in this environment (output). Some agents can perceive-and-act directly and others can reason about the perceived situation, plan, decide and then act. Another important property of the agents is their **autonomy**, since agents usually should be able to act without a continuous external control. In this section we will introduce some concepts about agents, behaviour and autonomous intelligent behaviour, according to [Osorio 2005].

From a more general point of view, Intelligent Agents approach has been broadly investigated and applied into several areas of the Computer Science, such as Artificial Intelligence (AI), Computer Networks (CN), Information Systems (IT) and Computer Graphics and Virtual Reality (CG/VR). As one of the central approaches studied in the modern AI, quickly it was adopted in other areas. This interdisciplinary characteristic reflects the current state of the research in this field, where there is no single and "perfect" definition or even a common sense about what is an agent and which are its main properties and desired abilities. Each research group has its own definition of an Intelligent Agent according with their objectives and applications, even if some points of these different definitions are common: Goodwin (1994); Maes (1994); Hayes-Roth (1995); Russell and Norvig (1995); Wooldridge and Jennings (1995); Franklin and Graesser (1996); Tecuci (1998).

In VREs (Virtual Reality Environments), agent based approaches are largely explored, and they are becoming one of the central pieces of such applications. Some of the main definitions for the "agent" term are presented in the next section, its properties and classifications, its modelling and implementation methodologies, as well as the main agent control architectures.

IV-1. Agents: Definitions and Properties

Russell and Norvig (1995) define an agent as a system capable to **perceive** information from the environment through sensors, and to **react** through actuators. According to Tecuci (1998), an agent is a knowledge based system that perceives the environment where it is immerse (e.g. virtual world objects, other agents and also the user actions); reason in order to interpret perceptions; solve problems and determine actions; and finally acts over the environment elements in order to accomplish a group of tasks for which it was designated. As defined by Garcia and Sichman (2003), Intelligent Agents are computational characters that act following a script, directly or indirectly defined by a user. According to Wooldridge and Jennings (1995), they possess certain fundamental properties, like: **autonomy** (they can operate without human direct intervention or from the other agents, and they have some control about their own actions); **social ability**

(they interact with other agents – humans/avatars or computational agents - or interact with the environment elements, depending of their necessity to solve problems, or even to help other agents); **reactivity** (they perceive the environment and they react to the modifications on it); **proactive** (they can exhibit a behavior driven by goals, making decisions and actions when they judge appropriate). In the virtual agents context, especially virtual humans, Thalmann and Thalmann (1994) indicate certain properties that should be considered in the agents' modelling: **behaviour** (the way the agent acts in the environment); **perception** (the agent observations, obtained from "physical" sensors); **memory** (represents the recovery process of what it was learned, especially based on associative mechanisms); **emotion** (defined as an effective aspect of conscience, a feeling state); **conscience** (state that characterizes sensations, emotions and desires); and **freedom** (characterized by the absence of restrictions in the agent choices or actions).

IV-1-1. Perception

So as to act in the environment, a virtual agent should perceive it: this perception includes the interaction between an agent and his environment (objects), the agent and the user of the system, and also between the agent and the other virtual agents. The perceived objects by the agent can be static objects (e.g. walls, furniture, obstacles, etc), other agents (static or moving agents), and dynamic obstacles (moving objects, like cars and doors), where the moving obstacles can be predictable (they move using predefined routes) or not. The mechanisms used to implement the virtual perception may vary, and include techniques like: line-of-sight (intersection within the agents' visual field, usually represented by a line, and a target object); contact detection (based on collisions and "bumper sensors"); measures obtained by distance sensors (like a sonar or an infrared sensor); alarms activated when entering the fields of action of some other object or agent; up to more complex biologically inspired models of vision, touch and auditory systems. Many of those perception techniques and sensors are simulated versions of some methods and sensors used in mobile robotics [Dudek 2000]. The agents can also memorize their perceptions and maintain updated a symbolic model of the environment (map), from which they can reason and plan actions.

IV-1-2. Action and Behaviour

From a more general point of view, the actions of a virtual agent are directly related to its movements and physical interaction with the other virtual world elements. The complexity of this action varies according to the sophistication of the activities performed in the environment, and the possible interactions with objects, other agents or users of the system. It is important to highlight that in the animation context (geometry), the term behaviour differs from the adopted in the Artificial Intelligence (AI) domain. In Computer Graphics, this term usually refers only to a predefined animation, while in AI it involves mainly the occurrence of events in the environment that lead to a specific reaction (reactive behaviour).

In general, computer animation techniques involve a predefined repertoire of animations, which is applicable when the agent actions are repetitive, or in other words, when the agent doesn't needs to have an autonomous control. According to Aylett and Luck (2000), considering that the virtual agents have a realistic corporal structure, is an

important aspect to provide an autonomous control on it, with the agent's movements determined by internal activations and not only as a result of an external animation of their corporal surface. In this context, the virtual agent's physical actions involve not only geometric structure calculations for the animation, but also a high level autonomous control, associated to the semantics of the actions in the environment. This approach is adopted by several behavioural animation researchers (Reynolds, 1987; Terzopoulos et al. 1994). According to Parent (2002), the behavioural animation consists of modelling elements that obey certain behaviour rules.

One of the first works in this area of behavioural animation was from Reynolds (1987), which simulated the behavior of virtual birds ("boids"), as described in section III-1. According to this author, complex movements can be modeled starting from a group of simple rules, associated to each agent, such as maintaining a minimum distance from obstacles and a certain movement speed. Following this approach, Terzopoulos et al. (1994, 1999) presented the simulation of virtual fish groups, where each fish is endowed with perception (artificial vision), locomotion control (based on a mass-spring system used to propel the fish in the water) and with behaviors (based on a group of parameters, such as hunger degree and predators fear).

We can also cite other specific approach based on behavioural animation, which is also based on physical models, used in the behavioural simulation of virtual humans' groups in emergency situations, proposed by Braun et al. (2003). This approach is based on a model that considers individuals' characteristics in emergent groups, consisting of a generalization of the physical model of crowds simulation proposed by Helbing (2000).

Concluding, as discussed above, virtual agents usually will need a complex control of their actions (including characters animation), that should consider the environment perception/interaction and also their objectives. This imposes the necessity to define a strategy and/or control architecture, that will be responsible for the agent's actions control. This control architecture will also be responsible for monitoring external events that happen in the environment and can affect the agent decisions and actions. Some classical autonomous and semi-autonomous agent control architectures are discussed in the next section.

IV-2. Control Architectures

An agent's architecture is a methodology adopted for the implementation of the cognition and decision process of this agent. Through this architecture, it is specified how an agent behaves during the interaction with the environment and how he acts in the accomplishment of his tasks. From an AI perspective, the agent control architectures can be classified in: reactivate, deliberative, and modular or hybrid. In the following sections, these architectures are presented.

Reactive Architecture. The control architecture is denominated reactive or non-deliberative when the choice of the action to be executed is related strictly and directly with the occurrence of events in the environment. In this architecture, the agent's actions control is accomplished based on behaviours of type perception-action (or stimulus-response). The agent acts almost immediately to the moment he perceives some special

"sign" coming from the environment, and his acts should be based only in this information provided by his perceptive sensors.

In this type of control architecture there is no explicit knowledge representation about the environment. The agents' knowledge is "implicit" and it is manifested through their reactive behaviours. This can restrict the agent's autonomy and his capacity to learn and to improve the way he acts and behaves. Another outstanding characteristic of this architecture is the absence of past actions memory, resulting on the fact that the last actions doesn't influence directly to the next actions.

These agents based on a reactive architecture are denominated reactive agents or non-deliberative agents, and they don't possess high level reasoning or planning capacity. Because that, they are considered simpler entities than the cognitive agents. They are simple agents based on simple behaviours, with their actions defined according to the current state of the environment, or more specifically, according to their perception (sensorial input) of the environment state. In order words, events originated in the environment can fire agent actions.

As examples of this kind of reactive agents, the *boids* proposed by Reynolds [Reynolds 1999] present some simple reactive behaviours, which can be combined (see bellow Hybrid and Modular Architecture) to obtain more complex behaviours. We can also cite some simple (autonomous) robots that adopt controllers based on a reactive architecture [Medeiros 1998, Brooks 1991]. These robots possess a certain number of sensors that allow them to perceive the environment, and using this information they can activate their motors executing an action in the environment. Whenever a sensor detects an obstacle in the robot trajectory, the control architecture will generate an order to change the robot's movement direction or to stop it. This is the typical example of a "perceive- action" situation. In some situations, virtual autonomous agents can act like reactive robots, exploring the environment and avoiding obstacles.

Deliberative Architecture. The control architecture is denominated cognitive or deliberative when the choice of one action to be executed by the agent is obtained from a symbolic model of the environment in conjunction with a plan of actions used to achieve a goal. This architecture is based on the generation of a sequence of actions (plan) that are executed in order to reach a specific objective. These actions are based on the knowledge and some hypotheses the agent possesses about the environment, the other agents and the way he needs to act to achieve his main goal. In order to do that, an explicit knowledge representation of the environment is maintained, as well as the agent can keep in his memory a list of the last executed actions.

However, the deliberative architecture is typically unable to act fast and appropriately when faced to unexpected situations. It adopts the hypothesis that the world state remain static while the agent is executing his actions or processing some information to deliberate about the next actions. On the other hand, those architectures can explicitly represent goals and environment descriptions (e.g. environment map), being able to reason, plan, act and even to learn (adapt the internal knowledge base of the agent).

The agents based on a deliberative architecture are denominated, cognitive agents or deliberative agents, and they reason and decide about which objectives they should reach, what plans they should proceed and which actions should be executed in a certain moment. In this way, a deliberative agent executes an "intelligent action" when, possessing a certain objective and the necessary knowledge indicating that a certain action can conduct him to the accomplishment of this goal, he selects and executes this action.

The deliberative agents act based on their knowledge, because they have a knowledge base and high level reasoning skills. The agent knowledge base is directly related to the application domain and it is also related to the specific kind of interactions exist between the agents and the environment. Deliberative agents can also plan future actions based on the memory of past actions and accomplished tasks. So, deliberative virtual agents can represent in their memory information like: maps of the environment, specific predefined paths in the environment, predefined plans of actions used to accomplish specific tasks; but they can also reason in order to: establish a new route from one position to another one in the environment, answer a specific user question, find a solution for a specific posed problem.

As examples of this kind of deliberative agents, we have the agents that use A* (AStar) algorithm to previously determine a trajectory planning [DeLoura 2000] or predefined behavioural scripts and rules [Osorio 2005], or even BDI (Belief-Desire-Intention) architectures [Osorio 2005] with no external inputs (no perception). The deliberative architecture is largely adopted to control NPCs (Non Player Characters) in computer games, but usually some kind of perception is also necessary.

Modular, Hierarchical or Hybrid Architecture. The control architecture is denominated hybrid when the choice of one action is accomplished using a combination amongst the techniques used in deliberative and reactive architectures [Dudek, 2000]. This architecture was proposed as an alternative to solve the main deficiencies of the two previously described architectures. The agent based on a pure deliberative architecture is typically unable to react fast and appropriately when faced to new and unexpected situations. The agent based on a pure reactive architecture is unable to discover new alternatives and execute complex tasks that are different from those accomplished by its initial objectives and behaviour, as he doesn't possess reasoning and planning abilities.

The main objective of a hybrid architecture is to build an agent including at least two subsystems: a deliberative system, that contains a symbolic model of the world, used in planning and decision making, and a reactive system, that is able to react to unexpected events that can happen in the environment. The hybrid agents are usually designed using a hierarchical architecture. The lowest level in the hierarchy is represented by the reactive system and it is used to acquire information (and react immediately if needed) from the environment, from the other agents or from other sources. The deliberative components, responsible for planning, goals determination and high level reasoning, are used in the highest levels of the hierarchy. So, hybrid virtual agents can plan, and execute tasks and also react to different kind of situations.

IV-3. Multi-Agents Systems

Some constraints and specific behaviour simulation techniques arise when we deal with multi-agents systems (MAS) and crowds of virtual characters, which are different from the modelling of virtual environment with no population at all, or populated with creatures that only act individually. Indeed, there are different types of techniques to be adopted depending on the different applications. First of all, the requirements for each MAS application are dependent on the nature of application. We use a classification of crowd application based on different purposes: Entertainment, Crowd Motion Simulation, Populated Collaborative Virtual Environments and Behavioural Modelling of Crowds [Osorio 2005]. These aspects are discussed in more details in [Osorio 2005], and also some important discussions about MAS behaviours (e.g. communication, cooperation – task sharing and planning, coordination) are addressed in [Weiss 1999].

In this text we will focus our attention to the behavioural modelling of crowds and the implementation of simulated crowds of agents. As a simple example of this kind of collective behaviour of crowds of agents, Reynolds [Reynolds 1987, 1999] described a distributed behaviour model for simulating *flocks of birds* formed by actors endowed with perception skills. In fact, the birds (or '*boids'*) maintain proper position and orientation within the flock by balancing their desire to avoid collisions with neighbours, to match the velocity of neighbours and to move towards the center of the flock. Reynolds's work shows real-time realistic animation of groups by applying simple local rules within the flock structure.

Musse & Thalmann [Musse 2001] proposed ***ViCrowd***: a model to describe crowd behaviours allowing the virtual agents to interact among them as well as with the Virtual Environment. ViCrowd's goal is to simulate groups of autonomous agents endowed with different levels of autonomy. This model is based on a hierarchical model to describe crowds with different levels of control: from guided to autonomous ones [Musse 2001]. The behaviour of crowds is based on rules dealing with the information contained in the groups of individuals.

A. Braun et al. [Braun 2005] also proposed a model to simulate virtual crowds in emergency situations, which was initially based on physics and socio-psychological forces in order to describe the human crowd behaviour in panic situations. This model also integrates the individual characteristics and depends on the group structure (since agents can be part of a social group and associated to it).

When more complex environments are adopted and the crowd behaviours are multiple and directly related to the different environment configurations, more sophisticated models of knowledge representation and manipulation are required. One example of this approach was proposed by D. Paiva et al. [Paiva 2005, 2005a], where an ontology is used in order to model the virtual environment, and also to direct the agents behaviours. This work will be described in more details in the next section and in the section VI-2 (practical applications) of this text.

V. Ontology-based VR Simulation

This section describes a virtual environment, which uses high level knowledge and reasoning capabilities, using ontology-based VR simulations. The agents could have different behaviours, since they can reason based on the knowledge they have about the world where they are inserted.

The VR environments that make use of knowledge representation models (e.g. ontology, informed environments) [Paiva 2005, Farenc 1999] to describe the agents (e.g. emotional states, personality, personal profile) and the environment (e.g. special places, functioning rules, place profile) allow us to obtain more complex and interesting behaviours in a multi-agent system. It is important to preserve the agents' individuality, in order to increase the simulation realism, but at the same time we need to manage complex scenarios composed by a great number of agents. So, in a complex MAS environment, the use of knowledge and automatic reasoning tools can make possible to control and generate complex collective and individual behaviours.

For example, we can add information about the environment describing the opening hours of a store, or the security measures associated to some flammable liquids (e.g. do not smoke nearby this objects, or provoke sparks). On the other hand, the agents can also use this knowledge, so they are able to decide where to go and how to act. In the above example, depending if the store is now open or closed, the agent will change his actions according to the present status of the environment (change the present destination), or also, if the agent wants to avoid some danger and needs to go to a place where there are some flammable liquids, the first action to be executed should be to extinguish his cigarette. This example explains some possible applications of the high-level reasoning, when associated to an informed VR environment.

In the next section we describe an example of application that makes use of an ontology-based VR simulation: Crowd Simulation in Normal Life Situations.

V-1. Using Ontology for Crowd Simulation in Normal Life Situations

Research on behavioral modeling has mainly focused on aspects of virtual human control in order to realistically populate virtual environments (VEs). Well known applications that require virtual population are games development and simulation of urban environments. Some of the work in the area [Braun, 2003; Barros, 2004; Musse 2001] have as focus crowd simulation during hazardous events. Other approach is to deal with "normal behavior" of virtual people that occurs before such events in different moments of normal life. The goal of this section is to present UEM – A Urban Environment Model [Paiva 2005], where knowledge and semantic information needed to realistically populate VEs is described by using an ontology that is associated to space and time of simulation, functionalities and normal occupation of space, etc. This is a novel approach

through which a large number of virtual humans can populate a urban space by considering normal life situations. Using this model, the semantic is included in the virtual space considering normal life actions according to different agent profiles distributed in time and space. For instance, children going to the school and adults going to work at usual times. Leisure and shopping activities are also considered. Agent profiles and their actions are also described using UEM.

Previous work has presented different ways for controlling virtual humans in order to improve their ability to evolve in an autonomous way in a virtual environment (not necessarily an urban environment). Several authors agree with the concept of autonomous or ”intelligent” agent requirements: autonomous behavior, action, perception, memory, reasoning, learning and self-control [Bordeux, 1999 ; Chaves, 1997 ; Meyer, 1994 ; Wooldridge, 25]. Also, several methods have been developed in order to model autonomous agents: L-systems [Luigi, 1990 ; Noser, 1996 ; Smith, 1990], vision systems [Marr, 1982]; rule-based systems [Girard, 1990]; learning methods [Smart, 1994 ; Sutton, 1984], evolution [Luigi,1990], etc. Specifically concerning VEs, methods for building informed virtual scenes focused on urban context have been discussed [Donikian, 1995 ; Donikian, 1999 ; Farenc, 1999 ; Thomas, 2000 ; Thomas, 2000b]. Donikian [1997] has studied the animation and simulation of autonomous vehicles in urban environments. Farenc [2000] described a database model in framework to simulate virtual humans. In these previous work as well as in other approaches on the domain [Kallmann, 2001 ; Musse, 1999], the main goal is to remove part of the complexity regarding some complex tasks processed by the agent and to transfer this information to the VE. As a consequence, a virtual agent can ”ask” to the VE the position of a specific place, the best way to go to a position, the semantic of a place or a path, or other needed information to provide the agents’ evolution in the virtual environment.

In UEM, the agents are created on the basis of an ontology, which includes information of population profiles as well as information about the urban environment, in a way the knowledge about the general model of the VE can be respresented and used as a basis to the simulation. People (virtual agents) can be created based on statistical data, if it is available, but also, fictitious information can be used. Agents move and behave in the urban life according to the time that are related to their usual activities, as described in the ontologies. In this way, people move during “normal life” in a more realistic way, without a “random aspect”, which is common in major part of related work. For instance, at 10:00 AM, if a user wants to check what is happening in the school, he or she will be able to see the students inside of it. Why is this similarity to the normal life is important in a simulation? For crowd simulation in panic situations, the location and the action people are doing when the hazardous event occurs is very decisive to determine the perception and response of people. By considering this aspect, the simulation can be much more realistic. For example, action response of people during the day or night can be different, and it can be simulated if we are able to model the normal life of people.

Besides presenting the details of the model, we present its integration into a crowd simulator whose main goal is to provide realistic and coherent population behavior into urban environments. The results show that urban environments can be populated in a more realistic way by using UEM, escaping from the normal impression we have in such kind of system that virtual people are walking in a random way by predefined paths in the virtual space. In the sequence, we describe the ontology associated to the virtual environment.

V-2. UEM – Urban Environment Model

UEM was built as an ontology using the ontology editor Protégé2000 (http://protege.stanford.edu/). Ontologies are explicit specifications of concepts related to a knowledge domain. Their use allows the interoperability among different systems and they can help the integration of different aspects in various kind of simulation systems. In our case it is a powerful tool for describing the environment and the crowd to be simulated, allowing the specification of the activity of urban environments to be done in a semantically organized way.

This means that the organization of the data is clear to the user and it can be easily adapted and expanded. Figure V-1 presents the basic model. Tables 1 to 4 show the main attributes for some of the components of the environment.

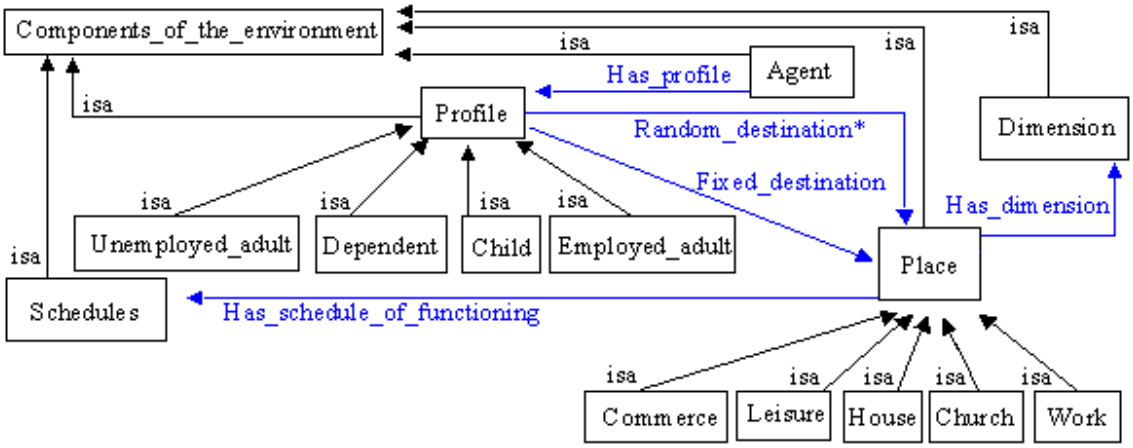

Figure V-1: Environment Ontology

Below we give an overview of the model, its main components, attributes defined in parenthesis ( ) and subclasses in brackets [ ]:

Agent (has profile)
A specific profile is assigned to each agent.

Profile (fixed destination, random destination)
[employed and unemployed adult, child, dependent]

Each profile will define their main activities, which corresponds to their usual (fixed) and eventual (random) destinations at certain times in the environment.

Place (has name, has capacity, has dimension, has schedule of functioning)
[leisure, house, commerce, church, work place]

Place refers to the agent destination according to their profiles. Places have capacity (number of allowed agents), and relations with schedule and dimension, the classes listed below.

Schedule (opening time, closing time, time of permanence, entering interval)

Dimension (coordinates X, Y, size)

Profile		
Name_of_the_profile	String	
Identifier_of_the_profile	Integer	
Fixed_destination	Instance	Place
Random_destination	Instance*	Place

Table 1: Profile properties

Schedules	
Opening_time	String
Closing_time	String
Average_time_of_permanence	String
Entering_time_interval	String

Table 2: Schedule properties

Dimension	
Y	Integer
X	Integer
Tamanho_Y	Integer
Tamanho_X	Integer

Table 3: Dimension properties

Place		
Name_of_the_place	String	
Capacity	Integer	
Identifier	Integer	
Has_dimension	Instance	Dimension
Has_schedule_of_functioning	Instance	Schedules

Table 4: Place properties

On the basis of this model, the activity of the population of the environment is defined. Children, for instance, will have as their fixed destination the school at a certain time with a determined time of permanence; at other times, they have random destinations in the environment. Besides children, the current model includes employed adults with fixed and random destination, unemployed adults with random destinations only, and dependents that only move when accompanied by an adult. By modeling the environment on the basis of this model we can generate a simulation that reflects more realistically the population activity in the environment. In the following section (VI-2) we describe in detail UEM´s implementation and its application on a real modeled environment.

VI. Practical Applications

In this section we describe two main different types of VR applications. The first one is related to the implementation of autonomous robots (SEVA and LegGen), which include the simulation of physical behaviours (kinematics, collision, sensorial perception and action in the SEVA Simulator; and articulated bodies affected by forces, gravity, friction and collision in the LegGen Simulator) and also the behavioural/cognitive control of these agents, using a hybrid architeture (SEVA) and evolutive strategies (LegGen). The LegGen simulation environment uses the ODE engine to implement the physics based simulation. The second type of VR application describes an ontology-based VR simulation (UEM), which implements high-level knowledge representation and reasoning for crowd simulation in normal life situations.

VI-1. Autonomous Robots

The VR simulation of autonomous robots is very important, since through the use of a realistic physical simulation tool, is possible to validate and test the design of different configurations of robotic platforms. In the case of the whelled autonomous robots, the physical properties of the virtual world should be correctly modeled, so the simulated sensors of the robots should correctly perceive the environment (e.g. edges, bumps) as the real sensors, and also the modeled vehicle kinematics should allow a correct simulation of the motor commands, resulting in a reliable simulation of the movement of robots. On the other hand, in the case of the legged robots, besides the sensors and actuators, the gravity and friction should also be correctly configured in order to test the robots. The Virtual Reality realistic simulation tools allows us to prototype and test the robots before to build the real ones, improving product development cycle success.

VI-1-1. SimRob3D simulator

We implemented a robot simulator that includes all necessary resources to realize experiments with autonomous mobile robots placed in a dynamic environment. This implementation was created in order to validate our proposed control system. This simulator was called SimRob3D (Mobile Robots Simulator based on Three-dimensional Environments) [Heinen 2002]. The main characteristic of this simulator resides in the fact that it implements our proposed control architecture using a three-dimensional environment for the navigation of the simulated mobile robots. The environment structure and objects can be designed with three-dimensional modeling software existing in the market (AutoCad, 3D Studio, among others), once we adopted in our simulator a common standard 3D data file format, the "3DS". This file format allows us to specify different elements composing the robot environment (walls, objects, light, textures), resulting in an environment with a high level of realism if compared with other environments used in some 2D based simulators [Heinen 2002].

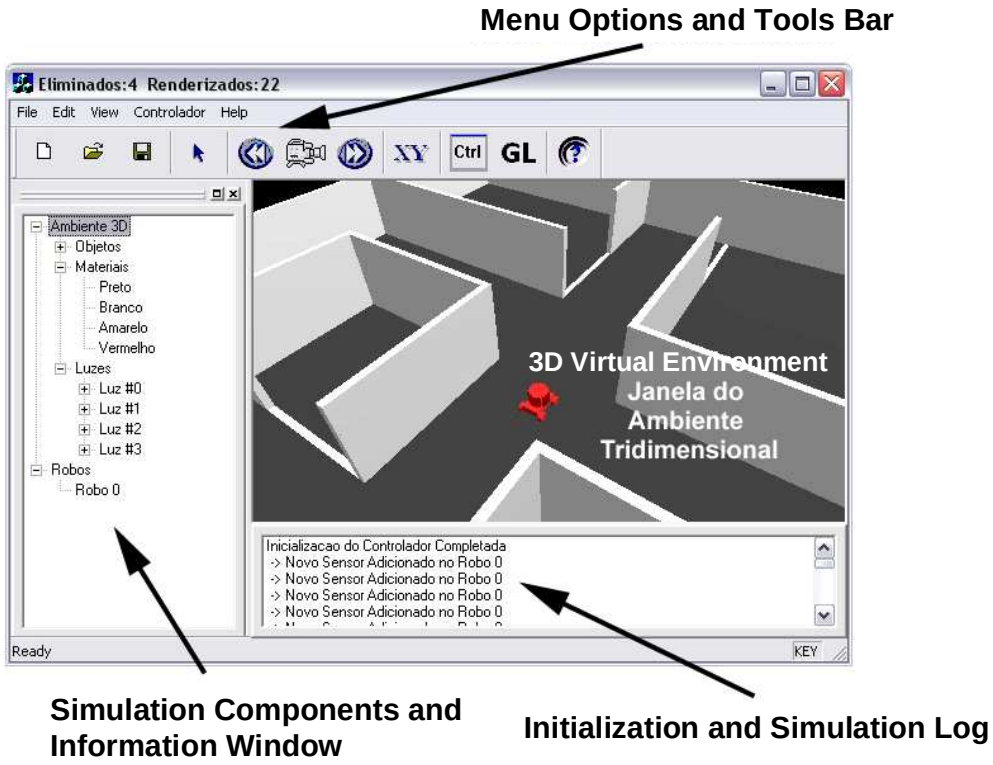

Figure VI-1: SimRob3D simulator

The simulator allows the user to place and configure obstacles, also allowing moving them in real time during simulation. The obstacles can also be pre-programmed to move in cyclical trajectories. The simulator possess several sensorial and kinematics models (e.g. encoder, sonar, infrared and laser sensors; Ackerman and differential kinematics), allowing the user to configure different types of robots. The Figure VI-1 shows the SimRob3D simulator.

All the sensors and actuators interact directly with the three-dimensional environment, becoming the simulation more realistic. Sensor and actuators are also modeled in a realistic way, i.e. we included in their model a gaussian error in order to introduce input sensor errors (e.g. noise) and output control errors (e.g. wheels sliding). These models of sensors and actuators provided to us a more realistic and non-deterministic robot behavior.

Another important characteristic of our simulator is its modularity. The controller is programmed separately from other simulation functions, implemented as a dynamic library (DLL). The controller is loaded at execution time, and it can be implemented using any language chosen by user. As the controller is completely separated from the robot and environment simulation, it has a well defined interface to communicate with the robot. The only information exchanged between robot controller and the other simulation modules are basically "Get_Sensors" and "Send_Command". So, we can easily replace the simulated robot by an actual robot [Heinen 2002].

VI-1-2. SEVA3D – Autonomous Vehicle Parking Simulator

Starting from the studies and research work developed by the Artificial Intelligence Group (GIA-PIPCA) and by Autonomous Vehicles Group (GPVA) at Unisinos[3] related to the development of applications in the area of mobile autonomous robots, the basis of this work were created [Jung 2005]. We should notice particularly the initial development of the SEVA3D[4] system (Autonomous Vehicles Parking Simulator in a Three-dimensional environment). Our main goal was to develop a system capable of accomplishing virtual realistic 3D simulations, implementing a better simulation tool of autonomous parking in parallel parking spaces. This system uses a realistic three-dimensional environment model with a 3D sonar sensor model, which includes noise as in real sonar data. Therefore SEVA3D is much closer to the reality, and the simulation results will be easily transposed to our automated Mini-Baja Buggy developed by the GPVA Group at Unisinos [Kelber 2005, Jung 2005].

The control system implemented in ***SEVA3D-A*** accomplishes the parking of vehicles, controlling them in an autonomous way, using a finite state automaton (FSA). The implemented system is also capable of pull out the vehicle of the parking space to go back to the traffic lane. The experiments using ***SEVA3D-A*** worked in a satisfactory way, but we needed to code manually all the rules used to define the FSA functioning. The task of specifying FSA rules has proven to be a difficult task, that needs a lot of a priori knowledge about the system functioning, and resulting a few robust vehicle control behavior in non expected situations.

For these reasons, we decided to develop a new version of the system SEVA3D, called ***SEVA3D-N*** [Heinen 2006d, Heinen 2006c]. The vehicle parking and pull out control in ***SEVA3D-N*** is accomplished using a neural network inspired in the Jordan-net model. The main advantage of this approach is the automatic knowledge acquisition instead of code it manually. The neural net is able to learn how to control the vehicle from examples, so we just need to show how to park and pull out the vehicle - showing how to do it, instead of specifying how to control it.

The ***SEVA3D-N*** Simulator possesses several improvements related to the ***SEVA3D-A***. The major advantage of ***SEVA3D-N*** is the fact that using an artificial neural network we do not need to manually code vehicle control rules (FSA), once the control system is obtained by training a neural net. We just need to show some examples of parking and pull out maneuvers execution and the system will automatically adapt the network parameters, learning to autonomously park and pull out the vehicle. The main components of the SEVA3D simulator are:

- *Perception Module*: measures the distance from obstacles based on a sonar sensor simulation;

[3] http://www.exatec.unisinos.br/~autonom/
[4] http://inf.unisinos.br/~osorio/seva3d/

- *Action Module*: sends action commands to vehicle actuators, defines the direction (forward, backward), the speed (gas pedal control) and the car orientation (steering wheel rotation);
- *Kinematics Module*: uses the Ackerman model to estimate the vehicle trajectory, considering vehicle direction, speed and steering wheel angle;
- *Control Module (Behaviour)*: receives sensor data from the Perception Module and sends actions to the Vehicle Action Module, using a neural network to control the parking and pull out maneuver.

The SimRob3D simulation tools [Heinen 2002] were used to create the virtual environment and to interface the vehicle devices with the SEVA3D autonomous controller implementation. The virtual environment was modeled, creating 3D models of the road and parked cars, and also the model of our autonomous vehicle. The Figure VI-2 shows an image of the virtual environment modeled.

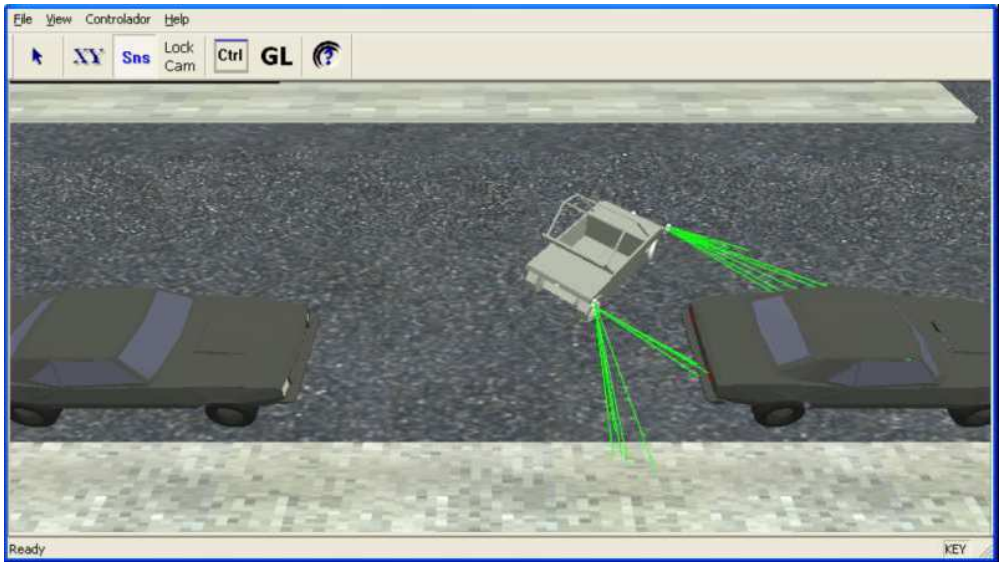

Figure VI-2: SEVA3D Virtual Environment Visualization

The model of the vehicle used to accomplish the parking task is a reproduction of a real Mini-Baja Buggy available in our research laboratory. This vehicle was developed by the GPVA Research Group at Unisinos. The real vehicle was automated and now it can be controlled from remote devices, like cell phones (see available videos in the GPVA web site). The Figure VI-3(a) shows the actual vehicle used as model of the virtual autonomous vehicle (Fig. VI-3(b)). At the present time we are working on the real vehicle instrumentation, adding sonar sensors, and a real world test is planned soon using the SEVA3D controller to control the real vehicle.

(a) Actual Vehicle **(b) Virtual Model of the Vehicle**

Figure VI-3: Automated Mini-Baja Vehicle

VI-1-3. LegGen simulator

The LegGen simulator[5] [Heinen 2006a, Heinen 2006b] was developed to accomplish the gait control of simulated legged robots in an automatic way. It was implemented using the C++ programming language and the free software libraries ODE and GAlib. The LegGen simulator works as follows: initially the file describing the robot is loaded, and the robot is created in the ODE environment according to file specifications. After this, the simulator parameters are loaded, and the Genetic Algorithm is initialized and executed until the number of generations is reached. The evaluation of each chromosome is realized in the following way:

- The simulated robot is placed in the starting position and orientation;
- The genome is read and the robot control FSM table values are set;
- The physical simulation is executed during a predefined time (60 seconds in our experiments);
- Fitness is calculated and returned to the GAlib;

During the simulation, if all paws of the robot leave the ground at same time for more than one second, the simulation of this individual is immediately stopped, because this robot probably fell down, and therefore it is not necessary to continue the physical simulation until the predefined end time.

Modeled robots

According to the documentation, computational complexity when using the ODE library is $O(N^2)$, where N is the amount of bodies present in the simulated physical world. Thus, in order to maintain the simulation speed in an acceptable rate, we should use few and simple objects. For this reason, all the simulated robots were modeled with simple objects, as rectangles and cylinders, and they have only the necessary articulations to perform the gait. Thus, body parts as the head and the tail are not present in the modeled robots.

[5] http://www.inf.unisinos.br/~osorio/leggen/leggen.html

In order to keep our robot project simple, the joints used in the robots legs just move around the *Z* axis of the robot (the same axis of our knees), and the simulations just used robots walking in a straight line. In the near future, we plan to extend our simulator to accept more complex robot models and joints. Several robot types were developed and tested, before we defined the final four main models presented here, that are shown in Figure VI-4.

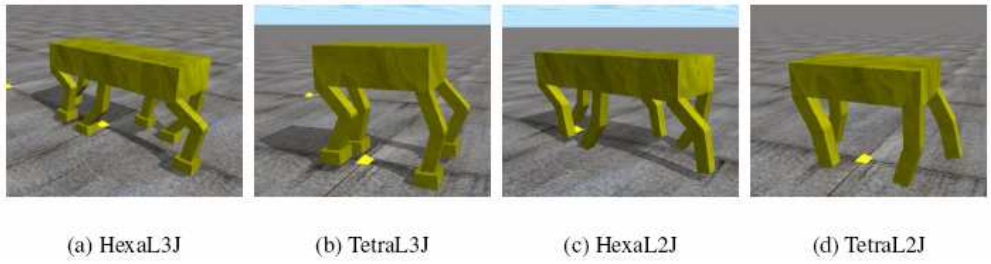

Figure VI-4: Robot models used in the simulations

The Table 5 shows the dimensions of the robots in meters. The simulated robots dimensions are approximately the dimensions of a dog. The use of paws as showed in Figures VI-4(a) and VI-4(b) models were designed to allow a more stable walk, mainly when dynamic stability was used. The robot joints have maximum and minimum joint angle limits similar to horses and dogs, but these animals have more articulated members than the implemented in our models.

	Body			**Thigh** and **shin**			**Paw**		
Robot	x	y	z	x	y	z	x	y	z
HexaL3J	0.80	0.15	0.30	0.05	0.15	0.05	0.08	0.05	0.09
TetraL3J	0.45	0.15	0.25	0.05	0.15	0.05	0.08	0.05	0.09
HexaL2J	0.80	0.15	0.30	0.05	0.15	0.05	-	-	-
TetraL2J	0.45	0.15	0.25	0.05	0.15	0.05	-	-	-

Table 5: Dimensions of the simulated robots

VI-1-4. Final Remarks

The LegGen experimental results of several different robot configurations are presented and compared in [Heinen 2006a]. These VR simulations are being used to improve the design of a real robot. The obtained results proved to be very helpful, allowing to compare different models of legged robots: with 4 or 6 legs, with or without large paws, robots evaluated only measuring the total distance walked and robots evaluated considering the addition of bumper sensors or gyroscope sensors (axis instability).

The SEVA3D simulator experimental results are also very helpful: they are being used to determine the optimal quantity and the ideal positions to place sonar sensors in the vehicle. The simulator is also very important in order to help us to develop our research about machine learning techniques (SEVA3D-N) used to implement autonomous vehicle control in parking tasks.

VI-2. Applying Ontology to a VR environment

This section describes a practical implementation of a VR simulation tool that uses ontology for crowd simulation in normal life situations.

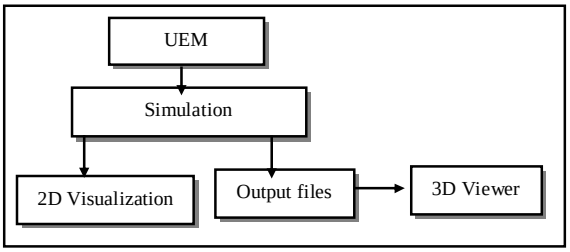

Figure VI-5: The Framework Visualization

VI-2-1. The Prototype of UEM

UEM defines the ontology of the virtual environment and the population configuration. The input information is processed in real-time and provides information to be visualized in 2D, as shown Figure VI-5. Moreover, generated data is exported in a proprietary format as input to PLAYER, which is a 3D Viewer based on CAL 3D library [CAL3D, 2005]. Player includes visualization of animated 3D life-like humans.

Figure VI-6 shows a screenshot of the simulator, and 2D visualization. We can see on the bottom-left of the image the time to be simulated (07:00 AM). At this time, major part of agents are at home. People will then begin to apply their activities, as defined on the ontologies. In Figure VI-7 we can see agents on the street, populating the environment, while figures VI-8 and VI-9 illustrate the 3D visualization.

Figure VI-6: At 7:00 AM people are at home

Figure VI-7: At 7:25 AM, people populate the VR Environment

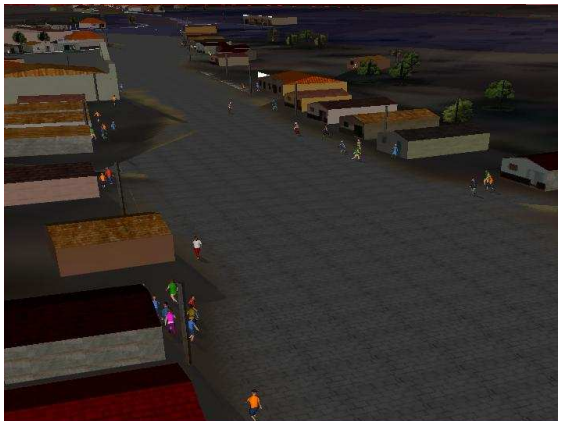

Figure VI-8: The 3D Viewer

For this simulation we considered São José, a village located in the Brazilian state of Rio Grande do Norte. The village has around 600 inhabitants, its main locations are two churches, a sports gymnasium (leisure), one school, three stores (commerce), and one industrial plant (work place) in which 200 people work. This village was chosen due to the availability of information as it has been used as case study for a system in a related project concerned with panic situations, PretoSim, which is briefly described below and fully presented in [Barros, 2004]. PetroSim was designed to help safety engineers to evaluate and improve safety plans and to help the training of people living in regions near to dangerous installations, the village of São José that is located near to petroleum extraction installations. The simulation considers different agent psychological profiles (dependent, leader or normal) and their behavior in danger situations.

Figure VI-9: The 3D Viewer

VI-2-2. Simulation Results

For the simulation on the basis of UEM, the user determines the distribution of profiles and people in the houses, and the environment is populated at random. The people move in the environment and they stay for a determined time interval in the places where their presence is required (obligatory) according to their profile (children at school, etc.). The movement is guided by the A*algorithm (general search algorithm that is used to solve. planning problems).

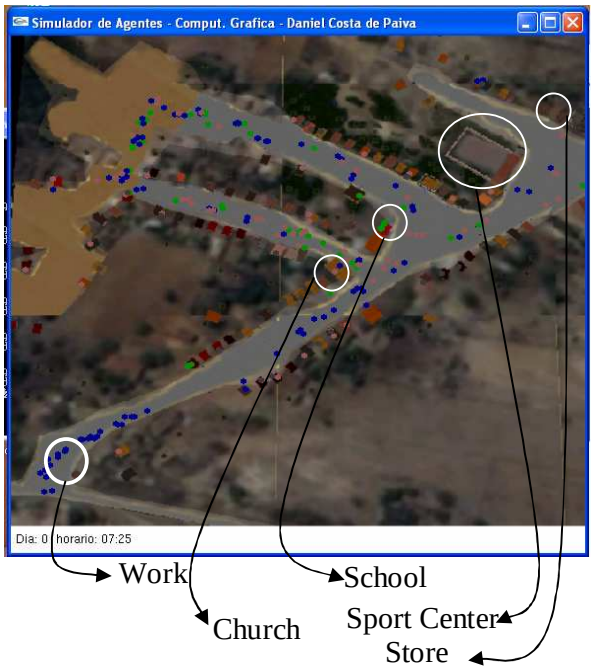

Figure VI-10: Locations of places

If the agent doesn't reach its place at regular time it doesn't stay and goes back home or any other nonobligatory place. Agents go home after 10 pm. Locations of some places defined on the ontology are represented in Figure VI-10.

Figure VI-11: At 11:29 AM, students and employed adults are in school and work, respectively. We can observe some other people on the street

The other images show the evolution of a specific simulation containing 250 people by using UEM. In Figure VI-11, at 11:29 AM, one can observe student and employed adults at school and work locations, respectively. Figure VI-12 shows the time students leave school and go home.

Figure VI-12: At 12:05 PM, students leave school and go to their home or other places, as considered in the ontology

The following graphs describe the spatial occupation of agents in Sao Jose Village, according to the simulation undertaken on the basis of our model. It is possible to observe that employed adults (Figure VI-13) spent more time at work, going to school after 07:00 PM (19:00), afterwards they adopt other behaviors, going to home, commerce, etc.

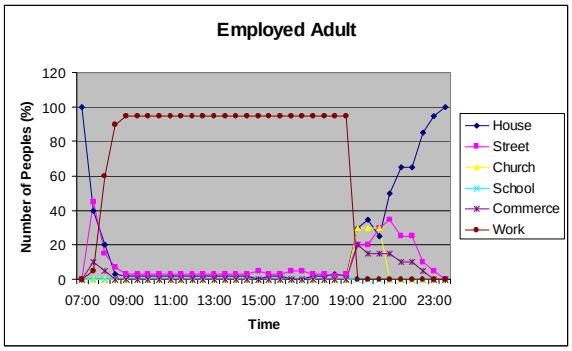

Figure VI-13: Spatial occupation of employed adults

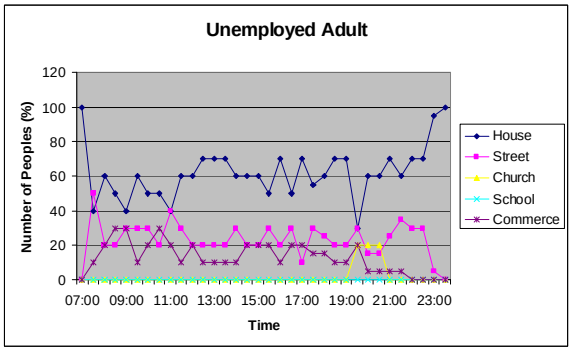

Figure VI-14: Spatial occupation of unemployed adults

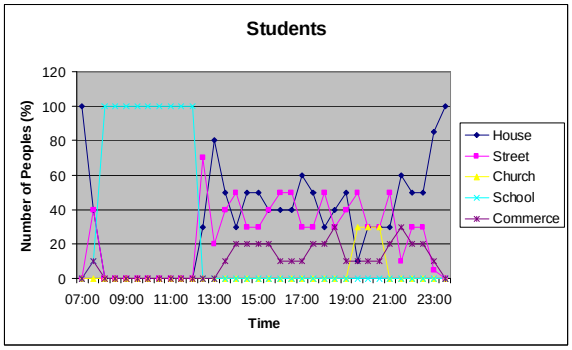

Figure VI-15: Spatial occupation of students

Students (Figure VI-15) stay at school until 12:00 AM, and then they adopt random behavior going to all possible places. After 10:00 PM (22:00) they stay at home. On the other hand, unemployed adults stay mainly at home, but populate other places too, as described in Figure VI-14.

The system allows user interaction during the simulation. The interactivity is given by the visualization of the graph, the description of ontology into the space, verification of profiles and etc. It is also possible to change specific place characteristics during the simulation, consequently changing the spatial occupation.

VI-2-3. Final Remarks

We have presented an ontology model and an ontology-based system that simulates normal life situations, as well as the results of its application to a small village situated in the state of Rio Grande do Norte, Brazil. The use of Artificial Intelligence techniques, such as knowledge representation through ontologies, results in an improvement of the quality of the result that pays off the complexity increase of the system.

The simulation we implemented presented 2D visualization of the normal life activity of 10K agents in São José Village achieved in real time frame rate (30 FPS) in a Intel® Xeon 2.40 GHz, 1GB RAM. UEM is a multi-plataform tool and can be executed under different operational systems, Windows® or Linux®.

With the simulation of São José village we were able to visualize the crowd activity and their movement in the space in a realistic way concerning the distribution of children, employed and unemployed adults in the village and the time of the day.

Using UEM, it is possible to extrapolate the number of people in Sao Jose village and determine the impact of it on the population patterns. For instance, if we know the capacity of specific service (e.g. school) we can estimate problems related to available space to build residences.

The model can be applied to other environments and can also be extended to support other types of agents and places. As future work we are considering the addition of features for the places and different, more detailed, profiles; testing the simulation of other environments.

Whereas here we mainly present normal life situation on its own, in the future, we plan to integrate this to panic situation, as already developed by the group [Braun, 2003 ; Barros, 2004, Musse, 2001], since the location and activity people are engaged when the hazardous event occurs is very decisive to the course of actions to take place. By considering this aspect, panic simulation can be much more realistic.

VII. Conclusion

The main goal of this tutorial was to present how important is the integration of physical and behavioural simulation techniques into Virtual Reality (VR) environments. We described the evolution of Virtual Reality tools, from the initial models of "VR visualization" to models of "VR cognitive" environments, which include tools used in visualization, physics simulation, autonomous agents behaviour simulation and control, and multi-agents implementation with advanced knowledge representation and reasoning.

The integration of these physical and behavioural simulation techniques into a VR environment can improve the user immersion experience into a virtual world, since it will be possible to obtain a more reliable reproduction of the real (concrete) world, as realistic as possible. We described some important tools and concepts used to integrate these simulation models into a VR environment. In order to demonstrate the practical application of these concepts, we presented two different types of VR applications, developed in our research group, which make use of physical and behavioural simulation techniques. The first set of VR applications presented are focused in the simulation and design of autonomous robots of two types: wheeled robots (SEVA3D tool) and legged robots (LegGen tool). The second type of application presented was a VR simulation tool that uses high-level knowledge representation and reasoning mechanisms (based on an ontology) to simulate crowd in normal life situations. All those applications demonstrate the importance and possible improvements obtained when physical and behavioural simulation techniques were integrated into virtual reality environments.

Acknowledgments

The authors would like to thank to F. Heinen, C. Kelber, C. T. dos Santos, FINEP-RBV, CAPES and Fapergs, for their support in the autonomous robots and autonomous agents research and devellopment work. The authors would like to thank to Julio Jacques Junior for the support with Player (UEM). This work was partially funded by CNPq. This work was also developed in collaboration with HP Brazil R&D.

REFERENCES

[AGEIA 2006] AGEIA PhysX - http://www.ageia.com/physx/ - Last Visited: 25/09/2006.

[Aylett 2000] Aylett, R. and Luck, M. (2000) Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. Applied Artificial Intelligence, v. 14, n. 1, p. 3-32.

[Baraff 1997] Baraff, B. and Witkin, A. (1997). Physically based modeling: Principles and practice. Online siggraph '97 course notes, Carnegie Mellon Univ. (CMU), Pittsburgh, CA.

[Barros 2004] L. M. Barros, A. T. da Silva, S. R. Musse. PetroSim: An Architecture to manage Virtual Crowds in Panic Situations. In CASA 2004. Geneva, Switzerland.

[Bordeaux 1999] C. Bordeux, et al. "An Efficient and Flexible Perception Pipeline for Autonomous Agents." *Proceedings of EUROGRAPHICS '99*. 18, (3), 23–29, (1999).

[Bourg 2002] David M. Bourg. Physics for Game Developers. O'Reilly Ed. 2002.

[Braun 2003] A. Braun, S. R. Musse, L.P.L de Oliveira and Bodmann, B. Modeling Individual Behaviors in Crowd Simulation. In CASA 2003 – Computer Animation and Social Agents. Pp. 143-148. May 2003, New Jersey, USA.

[Braun 2003] Braun, A.; Musse, S.; Oliveira, L.; Bodmann, B. (2003) "Modeling Individual Behaviors in Crowd Simulation". Proceedings of the Computer Animation and Social Agents, p. 143-148, May, New Jersey, USA.

[Braun 2005] A. Braun, B. J. Bodman, S. R. Musse. Simulating Virtual Crowds in Emergency Situations. In Proceedings of ACM VRST 2005 - ACM Symposium on Virtual Reality Software and Technology - Monterey, California, USA.

[Brooks 1991] Brooks, R. A. (1991). Intelligence without reason. Proceedings of the IJCAI - International Joint Conference on Artificial Intelligence, pages 569-595.

[Cal3D 2005] Cal3D Character Animation Library – http://cal3d.sourceforge.net. Accessed on January, 2005.

[Chaves 1997] A. Chaves, et al. "Challenger: A Multi-agent System for Distributed Resource Allocation." *Proceedings of Autonomous Agents'97*. ACM Press, 323–332, (Marina Del Rey, CA USA, Feb 1997).

[DeLoura 2000] Mark DeLoura. Game Programming Gems. Charles River Media. Vol. 1. August 2000.

[Donikian 1995] S. Donikian, E. Rutten. "Reactivity, Concurrency, Data-flow and Hierarchical Pre-emption for Behavioural Animation." *Paradigms in Graphics'95*, Eurographics collection. (Springer-Verlag, 1995).

[Donikian 1997] S. Donikian. ‘‘VUEMS: A virtual urban environment modeling system.” *Computer Animation ’97*. 127-133, (1997).

[Donikian 1999] S. Donikian. “Towards Scenario Authoring for Semi-Autonomous entities.” *International Conference on Visual Computing (ICVC99)*. (Goa, India, Feb 1999).

[Dudek 2000] Dudek, Gregory; Jenkin, Michael. Computational principles of mobile robotics. Cambridge: Cambridge University, 2000. 280 p.

[Enright 2002] Enright, D., Marschner, S., and Fedkiw, R. (2002). Animation and rendering of complex water surfaces. In Proc. SIGGRAPH, pages 736-744.

[Farenc 1999] N. Farenc, R. Boulic, D. Thalmann. “An informed environment dedicated to the simulation of virtual humans in urban context.” *Proc. Eurographics ’99, Computer Graphics Forum*, 18,(3),C–309–C–317, (1999).

[Farenc 2000] N. Farenc, et al. “A paradigm for controlling virtual humans in urban environment simulation.” *Journal Applied Artificial Intelligence*, 14,(1),69–91, (2000).

[Fedkiw 2001] Fedkiw, R., Stam, J., and Jensen, H.W. (2001). Visual simulation of smoke. In Proc. SIGGRAPH, Computer Graphics Proceedings, Annual Conf. Series, pages 15-22. ACM Press / ACM SIGGRAPH.

[Fedkiw 2006] Ron Fedkiw – Computational fluid dynamics and solid mechanics. http://graphics.stanford.edu/~fedkiw/ - Last Visited: 25/09/2006.

[Foley 1990] Foley et al. Computer graphics: principles and practice. Reading : Addison-Wesley, 1990. 1174p.

[Foley 1993] James D. Foley et al. Introduction to Computer Graphics. Addison-Wesley Professional, 1993.

[Foster 2001] Foster, N. and Fedkiw, R. (2001). Practical animation of liquids. In Proc. SIGGRAPH, Computer Graphics Proceedings, Annual Conf. Series, pages 23-30. ACM Press / ACM SIGGRAPH.

[Franklin 1996] Franklin, S. and Graesser, A. (1996) Is it an agent, or just a program? A taxonomy for autonomous agents. Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, Berlim.

[Funge 1999] Funge, John David. AI for Games and Animation: A Cognitive Modeling Approach. Natick, MA: AK Peters, 1999. 212 p.

[Garcia 2003] Garcia, Ana and Sichmann Jaime. Agentes e Sistemas Inteligentes (text in Portuguese). In: Rezende, Solange (Ed.). Sistemas Inteligentes: Fundamentos e Aplicações. Barueri: Editora Manole, 2003. 525 p.

[Girard 1990] M. Girard, S. Amkraut. “Eurhythmy: Concept and Process.” *The journal of Visualization and computer animation*, 1, 15–17, (1990).

[Goodwin 1994] Goodwin, R.. Formalizing Properties of Agents. Technical Report, School of Computer Science, Carnegie-Mellon University, Pittsburg, PA, September 1994.

[Hallam 1994] W. D. smart, J. Hallam "Location Recognition in Rats and Robots." Proceedings of Third International Conference on Simulation of Adaptive Behaviour. (Brighton, England, 1994).

[Hayes-Roth 1995] Hayes-Roth, B. An architecture for adaptive intelligent systems. Artificial Intelligence (Special Issue on Agents and Interactivity), v. 72, pp. 329-365. 1995.

[Heinen 2002] Heinen, Farlei J. and Osório, F. S. (2002). HyCAR - a robust hybrid control architecture for autonomous robots. In Proc. Hybrid Intelligent Systems (HIS), volume 87, pages 830-840, Santiago, Chile. IOS Press.

[Heinen 2006a] Heinen, Milton R. and Osório, F. S. (2006). Applying genetic algorithms to control gait of physically based simulated robots. In Proc. IEEE – WCCI Congress on Evolutionary Computation (CEC), pages 6714-6721. Vancouver, Canada.

[Heinen 2006b] Heinen, Milton R. and Osório, F. S. (2006). Gait control generation for physically based simulated robots using genetic algorithms. In Proc. Brazilian AI Symposium (SBIA), LNCS, Ribeirão Preto, SP, Brazil. Springer.

[Heinen 2006c] Heinen, Milton R., Osório, F. S., Heinen, F. J., and Kelber, C. (2006). Autonomous vehicle parking and pull out using artficial neural networks. In Proc. of I Workshop on Computational Intelligence (WCI), Ribeirão Preto, SP, Brazil. IEEE Press.

[Heinen 2006d] Heinen, Milton R., Osório, F. S., Heinen, F. J., and Kelber, C. (2006). SEVA3D: Using artificial neural networks to autonomous vehicle parking control. In Proc. IEEE WCCI Int. Joint Conf. on Neural Networks (IJCNN), pages 9454-9461.Vancouver, Canada.

[Helbing 2000] Helbing, D.; Farkas, I.; Vicsek, T. (2000) "SimulatingDynamical Features of Escape Panic". Nature, v. 407, pp. 487-490.

[Jung 2005] Jung, C. R.; Osório, F. S.; Kelber, C.; Heinen, F. Computação Embarcada: Projeto e Implementação de Veículos Autônomos Inteligentes. (Text *in Portuguese*). In: Barcellos, Marinho P.; Loureiro, Antônio A. F. (Org.). JAI - Jornada de Atualização em Informática / Congresso da SBC 2005. Porto Alegre, 2005, v. 1, p. 1358-1406.

[Kallmann 2001] M. Kallmann. "Object Interaction in Real-Time Virtual Environments." DSC Thesis" *DSC Thesis* , (2001).

[Kelber 2005] Kelber, C., Jung, C. R., Osório, F. S., and Heinen, F. J. (2005). Electrical drives in intelligent vehicles: Basis for active driver assistance systems. In Proc. IEEE Int. Symposium on Industrial Electronics (ISIE), volume 4, pages 1623-1628, Dubrovnik, Croatia.

[Luigi 1990] D. Luigi,V. Maniezzo. "The Rise of Interaction: Intrinsic Simulation Modelling of the Onset of Interacting Behavior" *Proceedings of First International Conference on Simulation of Adaptative Behavior*. MTI Press, (1990).

[Maes 1994] Maes, P. (1994). Modeling Adaptive Autonomous Agents. Artificial Life Journal.

[Marr 1982] D. Marr, Vision, Freeman Publishers. (New York, 1982).

[Medeiros 1998] Medeiros, Adelardo A.D. (1998). A Survey of Control Architectures for Autonomous Mobile Robots. JBCS - Journal of the Brazilian Computer Society, special issue on Robotics, vol. 4, n. 3.

[Meyer 1994] J. A. Meyer, A. Guillot. "From SAB90 to SAB94: Four Years of Animat Research." *Proceedings of Third International Conference on Simulation of Adaptive Behaviour*, (Brighton, England, 1994).

[Musse 1999] S. Musse, M. Kallmann, D. Thalmann. "Level of Autonomy for Virtual Human Agents." *Proceedings of ECAL '99*. (Springer, 1999).

[Musse 2001] Musse, S. R., Thalmann, D., " Hierarchical Model for Real Time Simulation of Virtual Human Crowds". IEEE Transactions on Visualization and Computer Graphics, vol. 7, n. 2, pp 152-164. April-June, 2001.

[Nguyen 2002] Nguyen, D. Q., Fedkiw, R., and Jensen, H. W. (2002). Physically based modeling and animation of fire. In Proc. SIGGRAPH, pages 721-728.

[Noser 1996] H. Noser. "A Behavioral Animation System Based on L-Systems and Syntetic Sensors for actors." *PhD Thesis*, EPFL. (Lausanne, Switzerland, 1996).

[ODE 2006] ODE – Open Dynamics Engine - http://www.ode.org/ - Accessed on September 2006.

[ODEWiki 2006] – ODE (Open dynamics engine) SourceForge Community Wiki Manual - http://opende.sourceforge.net/wiki/index.php/Manual - Last Visited: 29/09/2006.

[OpenSteer 2006] OpenSteer - Steering Behaviors for Autonomous Characters. http://opensteer.sourceforge.net/ - Last Visited: 25/09/2006.

[Osher 2002] Osher, Stanley J. & Fedkiw, Ronald P. (2002) Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag.

[Osorio 2005] Osório, F. S.; Musse, S. R.; Santos, C. T. dos; Heinen, F.; Braun, A.; Silva, A. T. da. Intelligent Virtual Reality Environments (IVRE): Principles, Implementation, Interaction, Examples and Practical Applications. In: Fischer, Xavier. (Org.). Virtual Concept (Proceedings - Tutorials). Biarritz, France, 2005, v. 1, p. 1-64.

[Paiva 2005] Paiva, D.; Vieira R.; Musse, S. Ontology-based Crowd Simulation in Normal Life Situations. Computer Graphics International – CGI 2005. Stony Brook, New York, USA. June 2005.

[Paiva 2005a] Paiva, D.; Tavares da Silva, A.; Vieira, R.; Musse, S. R. Modelling Gathering Events in Virtual Environments Using Ontologies. In Proceedings of V-Crowds 2005. Lausanne, Switzerland.

[Parent 2002] Parent, R. (2002) "Computer Animation: Algorithms and Techniques". Morgan Kaufmann. San Francisco. 527p.

[PhysX 2006] PhysX White Papers - PhysX Product Overview, Advanced Gaming Physics, Physics and Gaming. http://www.ageia.com/physx/ - Last Visited: 25/09/2006.

[Renault 1990] O. Renault, et al. “A Vision-based Approach to Behavioral Animation.” *The journal of Visualization and computer animation*, 1, (1),18–21, (1990).

[Reynolds 1987] Reynolds, C. “Flocks, Herds and Schools: A Distributed Behavioral Model”. Proc. SIGGRAPH ’87, Computer Graphics, v.21, n.4, July, 1987.

[Reynolds 1999] Reynolds, C. W. (1999) Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California. Pages 763-782.

[Reynolds 2006] Craig W. Reynolds – Boids and Steering Behaviors. http://www.red3d.com/cwr/ - Last Visited: 25/09/2006.

[Russell 1995] Russell, R.; Norvig, P. Artificial Intelligence: A modern Approach Englewood Cliffs, Prentice Hall, 1995. 932p.

[Smith 1990] A. R. Smith “Plants, Fractals, and Formal Languages.” Proceedings of First International Conference on Simulation of Adaptive Behaviour, MIT Press. (1990).

[Smith 2006] Smith, R. (2006). Open dynamics engine v0.5 user guide - http://www.ode.org/ode-latest-userguide.html - Last Visited: 23/02/2006.

[Sutton 1984] R. S. Sutton. “Reinforcement Learning Architectures for Animats.” *Computer Graphics*, 18, (3),1–10, (July, 1984).

[Tecuci 1998] Tecuci, G. (1998) “Building Intelligent Agents: A Multistrategy Learning Theory, Methodology, Tools and Case Studies”. Academic Press, 320 p.

[Terzopoulos 1994] Terzopoulos, D; Tu, X.; Grzeszczuk, R. (1994) “Artificial fishes with autonomous locomotion, perception, behavior and learning, in a physical world”. In Maes, P. and Brooks, R., editors, Proceedings of the Artificial Life IV Workshop, MIT Press, p. 17-27.

[Terzopoulos 1999] Terzopoulos, D. (1999) “Artificial Life for Computer Graphics”. Communications of the ACM, v.42, n. 8, p.33-42.

[Thalmann 1994] Thalmann N. and Thalmann D. (1994) “Creating Artificial Life in Virtual Reality”. Artificial Life and Virtual Reality, John Wiley, Chichester, pp.1-10.

[Thalmann 1999] Daniel Thalmann, Soraia Raupp Musse, Marcelo Kallmann. Virtual Humans' Behaviour: Individuals, Groups, and Crowds. Proc. Digital Media Futures, Bradford, UK. 1999.

[Thomas 2000] G. Thomas, S. Donikian. “Modelling virtual cities dedicated to behavioural animation. *Eurographics 2000,* (Interlaken, Switzerland, 2000).

[Thomas 2000a] G. Thomas, S. Donikian. “Virtual humans animation in informed urban environments *Computer Animation 2000*, IEEE, (Philadelphie, USA, May 2000).

[Ulicny 2001] B. Ulicny, D. Thalmann. “Crowd simulation for interactive virtual environments and VR training systems *Proc. Eurographics Workshop on Animation and Simulation’01*, 163–170, (Springer-Verlag, 2001).

[Watt 1993] Watt, A. 3D computer graphics. Reading : Addison-Wesley, 1993. 500p.

[Watt 2001] A. Watt and F. Policarpo. 3D Games: real-time rendering and software technology . Harlow: Addison-Wesley, 2001.

[Weiss 1999] Weiss, Gerhard (Ed.). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press: Cambridge, Massachusetts. 1999. 585 p.

[Wolff 2003] Wolff, K. and Nordin, P. (2003). Evolutionary learning from ¬ rst principles of biped walking on a simulated humanoid robot. In Proc. Advanced Simulation Technologies Conf. (ASTC), Orlando, FL.

[Wooldridge 1995] M. Wooldrigde, N. Jennings. “Intelligent Agents: Theory and Practice.” *Knowledge Engeneering Review*, 10, (2) Cambrigde University Press, (June 1995).