

# InViWo Agents: Write Once, Display Everywhere

Nadine Richard  
ENST - INFRES  
46, rue Barrault  
75634 Paris Cedex 13, France  
Nadine.Richard@enst.fr

## ABSTRACT

The INVIWO (*Intuitive Virtual Worlds*) platform enables the execution of virtual agents and avatars, whose behaviour can be described using the MARVIN synchronous language. In this paper, we focus on the visualization of INVIWO worlds; after a presentation of our model, we demonstrate how one can easily change the user interface to display the exact same scene, for instance to adapt to the capabilities of the terminal.

## Keywords

virtual agents, avatars, synchronous systems, Java3D

## 1. INTRODUCTION

When creating an inhabited virtual world, a key issue is the design of the autonomous entities that will populate the world and interact with the users. Through the INVIWO project, we aim at providing high-level tools for the design of such virtual worlds; we are focusing on the description of agent behaviours, by both visual and textual programming. We propose a specific language, called MARVIN, which is essentially based on reactive rules and on the primitive actions an agent can perform. This language enables the advanced user to easily combine various tasks such as goal-directed navigation and obstacle avoidance. The underlying agent model has been designed to present a minimal set of components, embedded in a fully dynamic architecture. We have been inspired by synchronous models to propose a deterministic and dynamic behaviour-based architecture for action selection. As an extension of the agent model, our avatar model clearly separates the avatar behaviour from the user interface, which enables the world designer to define various tools to display the same virtual world.

### 1.1 Related work

The reactive, bottom-up approach, which is strongly inspired by ethology and biology, has been successfully applied to behaviour-based robotics [1, 7], and to the animation and

simulation of virtual creatures, especially for navigation, obstacle avoidance, schooling and pursuit simulation [4, 15, 17, 19, 20]. As purely reactive agents lack of reasoning abilities and task-oriented behaviours, many authors have been advocating for hybrid agents that combine reactive and cognitive skills [1, 9, 13]. Those cognitive skills are necessary for chatterbots [11], artificial pets [4] or virtual humans [2, 18], that interact with human beings through gestures, facial expressions or natural language.

Finite-state automata have been extensively used to implement cognitive skills. Hierarchical parallel automata reduce complexity when designing concurrent behaviours [2, 12], but they are still hardly maintainable: a minor modification of the behaviour specification can lead to the complete redefinition of the automata. Specific languages have been designed for high-level behaviour scripting [2, 10], in particular for reviving the cognitive approach in the field of computer animation [8]. The PAT-NETS (*Parallel Transition Networks*), designed to simulate virtual humans, have been recently extended with an abstract action model to provide high-level description and parametrization of generic actions [2].

In virtual worlds, a user is usually represented by an *avatar* that allows her to explore the world and interact with objects, virtual creatures or other users. The text-based or graphical interface needed to manipulate the avatar is generally dedicated to one application, *e.g.* a video game, but there rarely are different interfaces available for the same virtual world. In the field of multi-agent systems or reactive robotics, the user is usually only an observer and cannot interact with the autonomous entities.

### 1.2 Organization of the paper

This paper is organized as follows: first, we describe the INVIWO model we have designed, *i.e.* the overall architecture we have chosen for inhabited virtual worlds, the model we have defined for autonomous virtual agents and the action selection architecture used by INVIWO agents. Then, in section 3, we extend our agent model to propose an original avatar model, where the avatar is clearly separated from the visualization interface. Section 4 is devoted to an example of virtual world described in MARVIN; we explain how to display different views of the same world. Before concluding, we briefly present the available INVIWO toolkit.

## 2. THE INVIWO MODEL

An InViWo world is a homogeneous multi-agent system, which evolves depending on the agent interactions, creations and destructions. The environment of an agent is just the set of the other agents. The agents communicate with each other by exchanging typed messages, called *stimuli*; interaction with the environment is thus achieved only through message-based communication. InViWo agents are fully autonomous: the internal state and the behaviour of an agent cannot be directly manipulated by an external entity (a scene manager, a user or another agent).

### 2.1 Architecture of the InViWo agents

An autonomous agent is a computational entity that is able to perceive its environment and act on this environment. It is autonomous since it decides what to do depending on the received stimuli, its own resources and its goals. The behaviour of an agent consists of a set of decision rules which lead to action selection.

As shown in figure 1, an InViWo agent is made of attributes representing its characteristics (*e.g.* its shape, position, or mood) and its knowledge (*e.g.* a personal map of the world), sensors to perceive internal and external stimuli, effectors to perform internal and external primitive actions, and a decision process.

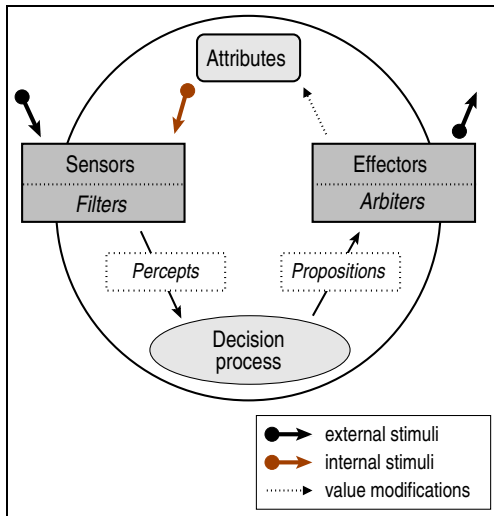


Figure 1: Basic architecture of an InViWo agent.

Effectors encapsulate abstract actions to be realized through combinations of the three available primitive actions: send an external stimulus, modify the value of an attribute or modify the agent structure (add or remove a component). An internal state modification, either a structure or an attribute change, causes the appropriate internal stimulus to be generated. Sensors can receive both external stimuli from the environment and internal stimuli from inside the agent.

### 2.2 A behaviour-based architecture

In behaviour-based architectures for action selection, the three steps of the agent (sensing, action selection and action) are distributed among basic, concurrent behavioural modules, usually called "behaviours". Those modules interact

to achieve particular tasks (eating, mating, avoiding predators, *etc.*), thus making the global behaviour of the agent emerge from their interactions. R. Brooks has proposed one of the first behaviour-based architectures, the so-called *subsumption architecture*, which consists in hierarchical layers of concurrent behaviours combined with inhibition mechanisms [7].

Our behaviour-based architecture is inspired by the principles of reactive synchronous systems, in particular from the ESTEREL language [3], the nets of reactive processes [6] and the synchronous objects [5]. The InViWo behaviours are reactive, concurrent units that should respond *instantaneously* to input signals (incoming typed messages) by emitting output signals (generated typed messages); it means that output signals should be produced in the same instant (timestep) as input signals. The outputs of a module can be routed to the inputs of other modules, by creating specific channels; this connection is possible only if the linked ports have the same type. This linking operation is very similar to the definition of VRML routes between nodes. As shown on figure 2, the percepts built by the sensors and the action propositions received by the effectors are actually signals.

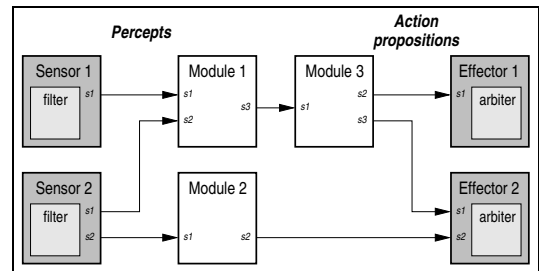


Figure 2: Example of a behavioural network.

A behaviour is basically a set of rules, which are triggered when the expected signals are received; activated rules lead to action propositions made to the effectors. Several rules can be activated at the same instant in the same behaviour. There is no assumption made on the level of complexity of a behaviour: one can easily combine reactive and cognitive skills, as both will be considered as black boxes.

Sensors and effectors are functionally similar to behaviours, except that they deal with both the synchronous network (inside the agent) and the asynchronous environment (outside the agent). The arbiter associated with an effector solves the conflicts between the requests from the concurrent behaviours when needed: the decision process of an agent is thus shared between the behaviours and the arbiters, as found in the DAMN architecture [16]. The effectors finally perform the chosen actions.

## 3. THE ROLE OF THE AVATAR

Within a virtual world, the human operator (or user) is usually represented by an avatar, which is in charge of two tasks: it represents the user towards the other users, and it includes navigation and basic perception capabilities. In most cases, the object representing the user can move within the world and a camera gives her a subjective view of the 3D scene. The simplest way to integrate users into InViWo worlds is

to consider the avatar as a specific agent, which both represents the user towards the other agents, and gives her a personal view of the environment with the capability to act on it. The INViWO avatar is thus a bidirectional translator:

- It receives external stimuli, and it informs its user about its internal state and about the environment.
- It executes user commands by modifying its attributes and structure and by sending external stimuli into the environment.

An INViWO avatar is then an agent that can be interactively controlled by a human operator. As shown in figure 3, the interaction between the user and her avatar is realized by adding specific stimuli called *user stimuli*. The user interface must be able to send and receive user stimuli in order to communicate with the avatar.

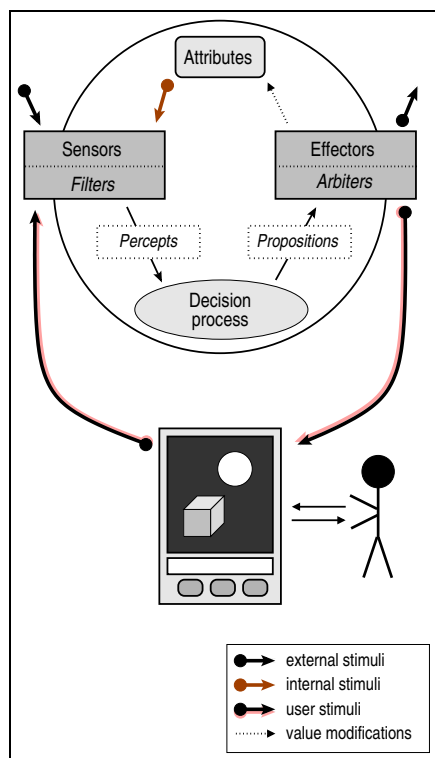


Figure 3: The InViWo avatar and user interface.

This model has two main advantages:

- The avatar is still fully autonomous but it can take user commands into account. User and external stimuli are processed the same way by the sensors, and the conflicts between the autonomous part and the "user-controlled" part of the avatar are homogeneously solved by the arbiters. A designer can reuse behavioural components defined for an autonomous agent to obtain an avatar able to avoid obstacles by itself.
- The user interface is completely separated from the avatar and from the execution platform. Of course,

the interface is dependent on the type and format of messages that are exchanged with the avatar, but we introduce a level of abstraction that allows designers to implement various user interfaces for the same INViWO world.

Moreover, this model allows us to clearly separate the "logical" and the "graphical" behaviour when designing an agent. The logical behaviour is described in MARVIN, while the graphical behaviour depends on the user interface and can be defined in various formats (VRML or 3DS file, JAVA3D code, etc.).

#### 4. USING THE MARVIN LANGUAGE

The MARVIN language was inspired by ESTEREL, with the addition of data manipulation facilities and agent-oriented features. It enables the modular programming of agents and agent components, according to the synchronous, behaviour-based architecture we have previously described.

Behaviours implemented in MARVIN are made of an interface, a set of local definitions (functions and variables) and a body. The interface part contains the declarations of both input and output signals. The body part contains a composition of imperative and reactive instructions. A behaviour can only act by modifying its internal state or emitting output signals. Sensors and effectors are syntactically very similar to behavioural modules.

The following excerpt of a MARVIN program shows how we can describe a simple obstacle avoidance behaviour:

```
behaviour avoid_obstacle
in Warning ;
out Displacement (float, vector_3D) ;

var weight <- 2.0;
var alpha <- 0.7854 ; // Avoid by the left.

body
loop
wait
Warning ->
var angle <- #orientation + alpha ;
var displ <-
new vector_3D (cos (angle), sin (angle), 0.0) ;
displ <- multiply_v3D (displ, #step) ;
emit Displacement (weight, displ) ;
weight <- weight + 0.5 ;
else
Tick ->
weight <- 2.0 ;
end wait
end loop
end body
end behaviour
```

This behaviour continuously waits for the pure **Warning** signal, which is supposed to inform the behaviour that a collision risk has been detected. If this signal is received, then the behaviour emits a weighted proposition of displacement. This vector is computed using the current value of the **step** and **orientation** attributes of the agent owning the behaviour; attributes are distinguished from local variables by the **#** prefix. If the **Warning** signal is not present, then the behaviour expects the **Tick** signal to be received. **Tick** is

a pure, predefined signal, which corresponds to the internal clock; it doesn't need to be declared in the module interface. As the Tick signal is always present, the weight of the proposition will be initialized if Warning is absent, which means that the weight grows only when successive Warning signals are received (at successive instants).

The following piece of code describes a simple robot, which detects and avoids agents representing walls:

```
agent my_robot
  attribute kind ;          // Type of creature.
  attribute position ;     // Absolute position.
  attribute orientation ;  // Absolute orientation.
  attribute speed ;        // Speed vector.
  attribute step ;         // Displacement step.

  var detect_walls ;      // Walls sensor.
  var explore ;           // Exploration behaviour.
  var avoid ;             // Collision avoidance.
  var control ;           // Position effector.

  function _init_ (x, y, z)
    kind <- "robot" ;
    position <- new position_3D (x, y, z) ;
    orientation <- 0.0 ;
    speed <- new vector_3D (0.0, 0.0, 0.0) ;
    step <- 0.05 ;

    explore <- new exploration (step) ;
    attach (explore) ;
    detect_walls <- new telemeter_sensor () ;
    attach (detect_walls) ;
    avoid <- new avoid_obstacle () ;
    attach (avoid) ;
    control <- new displacement_effector () ;
    attach (control) ;
  end function

  body
    link explore.Displacement control.Displacement ;
    link detect_walls.Warning avoid.Warning ;
    link avoid.Displacement control.Displacement ;
    esend E_kind (kind) ;
    esend E_robot_state (position, orientation) ;
  end body
end agent
```

In this excerpt, we first declare the agent attributes, especially `step` and `orientation` needed by the `avoid_obstacle` behaviour. We then declare the behavioural components (one sensor, two behaviours and one effector). The `_init_` function sets the agent attributes, and initializes the behavioural components. Note that a component must be “attached” to the agent to be executed. The body defines what should be done at the very first instant of the agent; in our example, it creates the needed channels between components and inform the environment about the initial state of the agent.

Finally, we can describe the initial state of the virtual world, *i.e.* instantiate the agents, the avatar and the user interface:

```
[...]
var robot <- new my_robot (0.0, 0.0, 0.0) ;
var avatar <- new my_avatar () ;
var uid <-
  create_user_interface
    ("InViWo.Examples.Ambience.RobotInterface2D") ;
attach_user (uid, avatar) ;
```

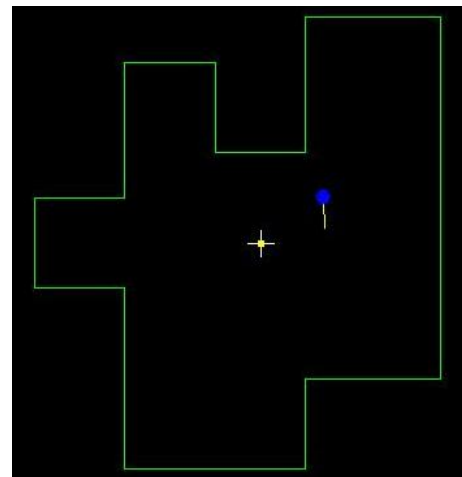


Figure 4: A 2D view for the robot.

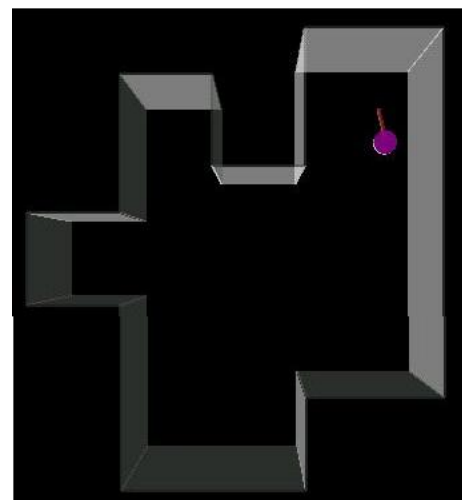


Figure 5: A 3D view for the same example.

```
launch_user_interface (uid) ;
[...]
```

The user interface should be attached to an avatar, in order to receive the user stimuli needed to visualize the world. Note that it can be attached to several avatars, in order to get information from different points of view. An avatar can also communicate with several user interfaces, and thus become a “multi-user avatar”.

Figure 4 is a snapshot of the running robot example, using a 2D user interface; the name of the interface is a complete JAVA path. To get a 3D view of the world, we only have to change this name: figure 5 is a snapshot of the same example, using the `RobotInterface3D` interface.

More robot simulations are currently being developed for the European AMBIENCE project. Visualization of the robot environment and behaviour should adapt to various terminal capacities (*e.g.* PDAs and laptops running GNU/LINUX). Many different situations and appropriate reactions have

been programmed. The result of the simulations helped to choose the optimal place of the infrared sensors on the physical robots.

The INVIWO execution environment is currently being ported onto the robots themselves. Therefore we will be able to use the MARVIN language to describe complex behaviours of individual robots acting in a community.

## 5. THE INVIWO TOOLKIT

The current prototype of the INVIWO toolkit is developed in JAVA, to be more portable and to take advantage of simple GUI design. The INVIWO platform is in charge of executing agent behaviours and handling the communication between entities (agents and user interfaces). We use the JAVA3D API for representing the world in 3D, and the SWING 2D API for the 2D view. The MARVIN parser has been implemented using the ANTLR library for JAVA [14]. The whole code and examples are available on demand as free software, under the GPL (*General Public License*).

## 6. CONCLUSION

Our multi-agent model for inhabited virtual worlds allows us to describe autonomous virtual creatures by aggregating basic components and behaviours into a deterministic, synchronous network. The MARVIN programming language and the INVIWO model make it easy to extend an agent into an avatar without encumbering the agent code. The same scene can be viewed and acted upon using different user interfaces, possibly at the same time. The user stimuli model we have designed will enable us to teleoperate autonomous physical robots living in a community using various devices, such as PCs, laptops and PDAs.

## 7. REFERENCES

- [1] R. ARKIN. *Behavior-based Robotics*. MIT Press, 1998.
- [2] N. BADLER ET AL. *Real Time Virtual Humans*. International Conference on Digital Media Futures, Bradford (UK), April 1999.
- [3] G. BERRY *The ESTEREL v5 language primer*. Technical Report, Centre de Mathématiques Appliquées (INRIA et École des Mines de Paris), 1999.
- [4] B. BLUMBERG, P. TODD, P. MAES. *No bad dogs: ethological lessons for learning in Hamsterdam*. From Animals to Animats 4, Cape Cod (USA), September 1996.
- [5] G. VIDAL-NAQUET, F. BOULANGER *Integration of synchronous modules in an object-oriented language* Information Systems - Correctness and Reusability, selected papers from the IS-CORE Workshop, World Scientific, 1995.
- [6] F. BOUSSINOT *Nets of reactive processes*. Technical Report, Centre de Mathématiques Appliquées (INRIA et École des Mines de Paris), 1994.
- [7] R. BROOKS. *Cambrian Intelligence: the early history of the new AI*. MIT Press, 1999.
- [8] J. FUNGE, X. TU, D. TERZOPOULOS. *Cognitive modeling: knowledge, reasoning and planning for intelligent characters*. SIGGRAPH'99, Los Angeles (USA), August 1999.
- [9] E. GAT. *Reliable goal-directed reactive control for real-world autonomous mobile robots*. Ph. D., Virginia Polytechnic Institute and State University, 1991.
- [10] B. LOYALL, J. BATES. *Real-time control of animated broad agents*. 15th Annual Conference of the Cognitive Science Society, Boulder (USA), June 1993.
- [11] P. MAES. *Artificial Life meets entertainment: lifelike autonomous agents*. Communications of the ACM, Special Issue on New Horizons of Commercial and Industrial AI, 38(11), November 1995.
- [12] G. MOREAU, S. DONIKIAN. *From psychological and real-time interaction requirements to behavioural simulation*. Workshop on Computer Animation and Simulation'98, Lisbon (Portugal), September 1998.
- [13] J. MÜLLER. *Control architectures for autonomous and interacting agents: a survey*. Intelligent Agent Systems (Theoretical and Practical Issues), Springer-Verlag, 1997.
- [14] T. PARR *et al.*. *ANTLR 2.7.1 reference manual*, October 2000.
- [15] C. REYNOLDS. *Steering behaviors for autonomous characters*. Game Developers Conference, San Jose (USA), March 1999.
- [16] J. K. ROSENBLATT *DAMN: a Distributed Architecture for Mobile Navigation*. Ph. D. thesis, Robotics Institute, Carnegie Mellon University, 1997.
- [17] N. SCHMAJUCK (Ed.). *Special issue on biologically-inspired models of navigation*. Adaptive Behavior, 6(3/4), Winter/Spring 1998.
- [18] D. THALMANN, H. NOSER. *Towards autonomous, perceptive, and intelligent virtual actors*. Artificial Intelligence Today, Lecture Notes in Artificial Intelligence, 1600, Springer, 1999.
- [19] O. TRULLIER AND J-A. MEYER. *Biomimetic navigation models and strategies in animats*. *AI Communications*, vol. 10, no. 2, 1997.
- [20] X. TU, D. TERZOPOULOS. *Artificial fishes: physics, locomotion, perception, behavior*. SIGGRAPH'94, Orlando (USA), July 1994.