Interactive Training of Virtual Agents

A.Del Bimbo, *Member, IEEE*, **E.Vicario**, and, **D.Zingoni**, *Member, IEEE*
Dipartimento Sistemi e Informatica, Universitá di Firenze
3, via Santa Marta, 50139, Firenze, Italy *

1 Introduction

Human-computer interaction has evolved from early textual programming towards visual languages with 3D direct manipulation interfaces. In this evolution, the cognitive effort of the user in managing objects of the system has been progressively reduced by augmenting its engagement in the operation. Virtual reality may be regarded as the ultimate stage in this trend [9].

In early textual interfaces, objects were not directly operated but rather the user conversed about them through an intermediate language[10].

In 2D direct manipulation interfaces, linear textual strings have been replaced through icons, i.e. synthetic images representing real-world objects or processes. Interaction largely relies on visual communication, and the user engagement is increased by the direct operation of object simulacres. In this operation, visual languages provide conceptual guidance in regarding spatial and temporal arrangements of icons as visual statements, and exploiting human natural capabilities in picture recognition and interpretation. Nevertheless, in 2D iconic interfaces, the dialogue between the user and the objects is still mediated by a language structure (albeit visual). This structure, though characterized by simple syntax and semantics, still requires a preliminary learning for effective operation of icons by the user.

In 3D direct manipulation interfaces, such as virtual reality environments, the world of interest is represented directly, and no intermediary exists between the user and the objects [8]. The interface features a set of virtual agents which change their state in response to external and internal stimuli so as to emulate some real-life environment. The user interacts with these virtual agents according to *syntactic* and *semantic* conventions which reflect the characteristics of the emulated environment. In this way, the need for a preliminary learning by the user is overcome by its knowledge of the emulated real-life environment.

To attain an effective emulation, virtual agents must exhibit both realistic graphic shapes and realistic behaviors. Realistic shapes help the user to perceive the virtual environment as a replica of a real context thus forcing ways of interaction that are usual in reality. Realistic behaviors improve the user's engagement in the virtual world, giving the sensation of interaction with a context populated by *living* and *natural agents*.

In this evolution, due to the increasing complexity of interaction systems, the use of appropriate notations, methodologies and tools supporting the designer during the entire lifecycle from the specification to the implementation stage assumes a major relevance. The development of virtual worlds is currently addressed by several ongoing projects, but only in few cases the design of agent behaviors is considered explicitly. The AVIARY system provides a general framework where the behavior of virtual agents is made up by *low-level* physical (rather than behavioral) laws [13]. In the VR toolkit [14], the specification of reactive behavior for virtual agents is supported through the use of rule-based specification.

Following this trend, at our Department we are currently involved in a project aiming at the development of a tool supporting an *evolutionary* approach to the interactive and visual specification of the behavior of virtual agents [5]. This tool enforces the concept of *behavioral training*, i.e. the interactive building of agent's behavior: each agent is natively provided with a set of *basic actions*; the behavior pattern by which the agent engages these basic actions in its interaction with the environment is specified by example through a visual interface. In this construction process, the behavior of each agent is defined by visually replicating its *reaction pattern* to specific *perceived stimuli* coming from

*This work was partially supported by MURST under project 40%.

the virtual world featured by the interface. The reaction pattern is represented through a Petri Net based model [12] which permits rapid prototyping so as to support interactive specification. Besides, spatio-temporal stimuli are represented through an original design language, referred to as eXtended Spatio Temporal Logic, which extends concepts of conventional Temporal Logic [11][1] so as to deal with space and time within a unified framework.

In the rest of this paper, the languages used for the representation of stimuli and reaction patterns are described in Sects.2 and 3. Afterwards, the visual interface of the training environment is presented in Sect.4 and conclusions are drawn in Sect.5.

2 Specification of Perceived Stimuli

Spatio-temporal relationships comprise the basic essence of interaction among the agents of a virtual environment. Gesture patterns (such as grasping, touching, pointing, approaching and the like) are largely pervaded by this type of relationships. According to this, in our approach, the agents of a virtual environment are sensitive to spatio-temporal stimuli, i.e. to the evolution over time of their spatial relationships with other agents in the environment.

The specification of this type of interaction model requires an appropriate language allowing for the expression of 3D spatio-temporal situations as encountered in the natural perceptual process of real-life agents. This requirement has a number of major implications:

- the language must encompass both spatial and temporal phenomena within a cohesive and homogenous framework;
- the language must be oriented towards the expression of *qualitative* spatial and temporal relationships so as to match the incomplete and imprecise knowledge which pervades the environment perception of real-life agents. Besides, the language must also incorporate some form of *smooth quantity* to permit the expression of *qualitative metric relationships* [7] such as: distances between agents (such as *near* and *far*); time intervals between subsequent conditions (such as *soon* and *late*); and relative speeds of the agents (such as *slow* and *fast*);

As to the expression of spatial relationships between objects, for each agent, spatial relationships must be referred to an individual reference systems, and metric conditions (both spatial and temporal) must be defined so as to match the characteristics of the individual perceptual process of the agent itself. To this end, each virtual agent must be provided with an individual spatial reference system and with individual spatial and temporal thresholds defining concepts such as *near*, *far*, *soon*, *late*, *slow* and *fast*; Both the individual reference system and the *qualitative metric thresholds* must be defined according to pragmatic criteria so as to reflect the geometric and behavioral characteristics of the agent itself.

To cope with these requirements, an original language, referred to as eXtended Spatio Temporal Logic (XSTL), has been designed, which extends the concepts of *conventional* Temporal Logic , so as to support a homogeneous description of both spatial and temporal phenomena. Specifically, the spatio-temporal evolution of the contents of a dynamic scene are captured through assertions that are organized in two nested *dynamic* and *static* levels [3]. At the dynamic level, *sequence assertions* expressed through the conventional operators of Temporal Logic capture the evolution over time of the spatial contents of the individual frames; In turn, these spatial contents are defined at the *static level* by *frame assertions* that are formed according to an original language which transposes the concepts of Temporal Logic so as to deal with space instead of time. By extending concepts proposed in [1] to augment Temporal Logic with quantitative modeling capability, XSTL introduces metric expressivity in both frame and sequence assertions so as to permit the representation of qualitative metric relationships among space points and time instants in which certain conditions hold.

2.1 Static Descriptions

The spatial relationships between the agents in a static virtual scene are expressed through *frame assertions* which transpose the concepts of conventional Temporal Logic so as to deal with space instead of time. In the following, the syntax of these assertions and their meaning are informally introduced. For a more rigorous treatment, the reader is referred to [6].

A generic frame assertion Φ is made up of a point r and a *spatial formula* ϕ: $(r) \models_s \phi$ and it is read as: "formula ϕ hods in point r". In turn, spatial formulae are formed by inductively composing the identifiers of the agents in the scene through the Boolean connectives, and a set of spatial operators which transpose in the space the semantics of the usual operators of Temporal Logic. Given a point r and an object p, the basic assertion of Spatial Logic is expressed as

$$(r) \models_s p \qquad (1)$$

and means that object p stands over r.

The spatial operators *eventually in the space* ($\Diamond_{e\pm}$), and *always in the space* ($\Box_{e\pm}$) permit to state a formula over some or all of the points encountered moving from a starting point r along the direction of a coordinate axis e_i. Specifically, the assertion

$$(r) \models_s \Diamond_{e_1^+}\phi \tag{2}$$

means that formula ϕ holds in *some* of the points that are encountered moving from r along the positive direction of axis e_1 Conversely, the assertion

$$(r) \models_s \Box_{e_1^+}\phi \tag{3}$$

means that formula p holds in *all* the points that are encountered moving from r along the positive direction of axis e_1.

The spatial operator *until in the space* ($unt_{e\pm}$), permits to state that a formula ϕ_1 holds over all the points encountered when moving from a starting point r along the direction of an axis at least until a point is encountered in which a second formula ϕ_2 holds:

$$(r) \models_s \phi_1 unt_{e_1^+} \phi_2 \tag{4}$$

In the construction of frame assertions, multiple spatial operators and Boolean connectives can be inductively combined so as to compose finer assertions. For instance, the assertion

$$(r) \models_s \Diamond_{e_1^+}\Box_{e_2^-}(p \vee q) \tag{5}$$

means that, starting from point r, a point r' is reached such that, in all the points encountered moving from r' along the negative direction of axis e_2, there are objects p or q.

The composition of *eventually* operators referring to opposite directions of the same axis permits to disregard the alignment condition orthogonal to that axis, thus permitting the expression of *don't care* conditions. For instance, the assertion

$$(r) \models_s \Diamond_{e_1^+}\Diamond_{e_2^+}\Diamond_{e_2^-}p \quad , \tag{6}$$

means that, moving from point r along the positive direction of the first axis ($\Diamond_{e_1^+}$) and along one of the two directions of the second axis ($\Diamond_{e_2^+}\Diamond_{e_2^-}$), a point r' is reached such that object p stands over r'. Of course, *don't care* conditions can be used for multiple axes; in particular, if this is done for all the axes, a *somewhere* condition can be expressed.

2.1.1 The Walkthrough Paradigm

Frame assertions deriving from the composition of multiple spatial operators can be regarded as spatial *walkthroughs* leading from a starting point r to a second point r' in which some spatial formula holds. In this perspective, the spatial relationship of an object p with respect to a point r can be represented through the set of the possible walkthroughs leading from r to any of the points in p. Walkthroughs are expressed considering paths parallel to the coordinate axes of an underlying reference system which, in our case, is assumed to be *object-centered.* With one such reference system, scene descriptions are independent of the observer's point of view. Each object is provided with its individual reference coordinate system, and the overall description of the scene is given by a set of descriptions, each capturing how one object *sees* the other objects with respect to its own reference coordinate system.
It is worth remarking that this *walkthrough paradigm* does not refer to object projections on coordinate axes, thus basically differentiating frame assertions of XSTL from previous approaches based on symbolic projections [2][4].

2.1.2 Context Declarators

The spatial relationship of an object p with respect to point r may be represented by the set of the possible walkthroughs from r to any of the points in p. By extension, the spatial relationship of an object p with respect to a second object q can be represented by the set of the walkthroughs from the points in q to the points in p. To this end, *context declarators* are introduced so as to refer frame assertions to the overall set of the points in an object: if q is an object identifier and ϕ a formula, the assertion

$$(q) \models_s \phi \tag{7}$$

(read "ϕ holds in q") expresses that, for each point r in q, there exists a walkthrough originating in r which satisfies ϕ. For instance, the assertion

$$(q) \models_s \Diamond_{e_1^+}\Diamond_{e_2^+}p \tag{8}$$

means that, for any point r in q, there exists a walkthrough along the positive directions of axes e_1 and e_2 which leads into a point r' belonging to p (see Fig.1a).

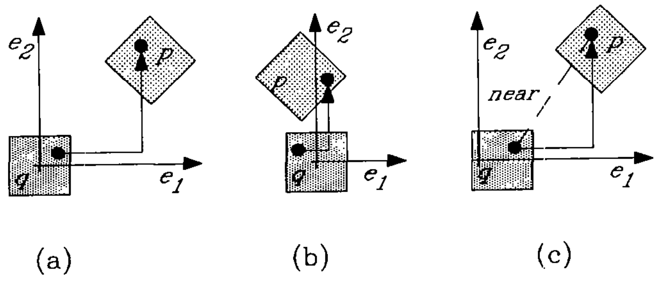


Figure 1: Three sample scenes.

It should be noted that, according to this definition, negation does not distribute with respect to context declaration. In fact, in general $(\neg(q) \models_s \phi) \neq ((q) \models_s \neg\phi)$. This permits the expression of assertions referring to *some* rather to than *all* the points in a context object q. For instance, the assertion

$$\neg\left((q) \models_s \neg(\diamond_{e_1^+}\diamond_{e_2^+}p)\right) \tag{9}$$

means that it is not the case that formula $\neg(\diamond_{e_1^+}\diamond_{e_2^+}p)$ holds for all the points in q, i.e. it means that there exists at least one point r in q such that the walkhrough $\diamond_{e_1^+}\diamond_{e_2^+}$ leads from r to a point in p (see Fig.1b).

The use of context declarations has a critical impact on the feasibility of a decision algorithm checking frame assertions for satisfaction. In fact, in order to check for satisfaction a frame assertion of the form $(q) \models_s \phi$, the formula ϕ must be verified for *all* the points of the object q, which may be not feasible if q stands over a dense set of points. This problem may be overcome if each object q is associated with a finite set of representative points, and the assertion $(q) \models_s \phi$ is read as "for any representative point r in q, the assertion $(r) \models_s \phi$ holds". Note that the selection of these representative points determines the model of objects assumed for the descriptions. and that different selections may be considered so as to accomplish different levels of detail and accuracy in the representation of the object relationships within a scene.

2.1.3 Restricted Fragments

In general, different XSTL assertions can be stated for a single scene. While giving means to description refinement in manual construction of assertions, this absence of univocality clashes with the possible need of automatic generation of XSTL assertions from the interpretation of sample scenes. To resolve this clash, some assumptions must be made about the kind of assertions generated by the system so as to obtain only one description for each scene. These assumptions involve the structure of Boolean composition of the different spatial operators as well as the number of objects considered in each assertion, and define the *fragment* of XSTL that will be actually exploited by the automatic generator of assertions.

As an example, a possible fragment is made up of the assertions which are formed through the use of *eventually* operators relating couples of objects. Using this structure of assertions, the spatial position of p with respect to q in Fig.2 can be expressed by two possible walkthroughs: $(q) \models_s \diamond_{e_1^+}\diamond_{e_2^+}p$ is possible for all the points in q, while $(q) \models_s \diamond_{e_2^+}p$ is possible only for the points of q in the dark-gray area of the figure.

Note that, saying that the spatial position of p is expressed by two walkthroughs means that: for each point in q, at least one of the two walkthroughs is possible and that each of the two walkthroughs is possible at least in one point of q.

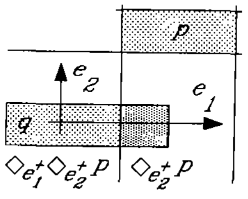


Figure 2: A sample scene allowing for two different walkthroughs from q to p.

2.2 Dynamic Descriptions

Static *frame assertions* can be composed through operators of conventional Temporal Logic such as *eventually* ($\diamond_t$), *always* ($\Box_t$) and *until* (unt_t) so as to form *sequence assertions* capturing the evolution over time of the spatial relationships expressed as frame assertions.

2.2.1 Sequence Assertions

A sequence assertion Θ is made up of a scene index j and a *temporal formula* θ: $(j) \models_t \theta$ and it is read as: "formula θ holds in the j^{th} scene of the sequence".

Temporal formulae are formed by inductively composing spatial assertions through the Boolean connectives, and the usual set of operators of Temporal Logic. For instance, if σ is a sequence of frames, j is a frame index and ϕ is a frame assertion, the sequence assertion

$$(j) \models_t \diamond_t\phi \tag{10}$$

means that, along the sequence σ, after the j^{th} frame, there is a frame in which the frame assertion ϕ holds. Besides, the sequence assertion

$$(j) \models_t \Box_t\phi \tag{11}$$

means that the frame assertion ϕ holds in all the frames subsequent to j along the sequence σ. Finally, the sequence assertion

$$(j) \models_t \phi_1 unt_t \phi_2 \tag{12}$$

means that, along the sequence σ, ϕ_1 holds in all the scenes subsequent to j at least until a frame in which ϕ_2 holds.

The appropriate composition of temporal operators permits the description of time ordering relationships between frames in which different spatial conditions hold. For instance, the sequence assertion

$$(t_a) \models_t \left((q) \models_s \diamond_{e_1^+}\diamond_{e_2^+}p\right) \wedge \diamond_t \left((q) \models_s \diamond_{e_1^-}\diamond_{e_2^+}p\right) \tag{13}$$

means that, object p is initially right of object q and that, eventually, it will be left of it.

It is worth noting that the above assertion permits any change of the spatial relationships between q and p before the final condition in which p is left of q is reached. This depends on the fact that the temporal *eventually* operator does not state a condition in the *next* state but rather in *some state in the future*. In general, the temporal *eventually* operator can be used to attain *partial* descriptions of a sequence in which only some relevant conditions are encountered. On the contrary, the temporal *until* operator permits the description of sequences in which *all* the subsequent conditions are defined.

2.3 Achieving Quantitative Descriptions

Following the approach proposed in [1], qualitative metric relations are added both in frame and sequence assertions through the use of spatial and temporal *freeze variables*.

2.3.1 Metrics in Spatial Descriptions

Spatial freeze variables are used to mark points on the objects, and to track their subsequent positions in the evolution of the sequence. For instance, the frame assertion

$$(r) \models_s v_r.\phi \tag{14}$$

means that $(r) \models_s \phi$ is satisfied if the position of point r in the current frame is assigned to the freeze variable v_r. After this freezing operation, the variable v_r cannot be used in any other freezing operation but it can be used to refer to the position taken by point r along the subsequent frames of the sequence.

Positional values assigned to spatial freeze variables may be used to express metric relationships regarding the *distance* between the points they track. For instance, the frame assertion

$$(r) \models_s \quad v_r.\left(\Diamond_{e_1^+}\Diamond_{e_2^+} v_{r'}.(p)\right) \tag{15}$$

means that, (see Fig.1c) starting from point r, whose position is freezed in the variable v_r, and moving along the positive direction of the axis e_1 and along the positive direction of the axis e_2, a point is reached, whose position is freezed in the variable $v_{r'}$, which is part of the object p. This assertion can be refined through metric conditions on the distance between the positions freezed in the variables v_r and $v_{r'}$:

$$(r) \models_s \quad v_r.\left(\Diamond_{e_1^+}\Diamond_{e_2^+} v_{r'}.(p)\right) \quad \wedge \quad (\|v_r - v_{r'}\| = near) \tag{16}$$

which specifies that the positions freezed in v_r and $v_{r'}$ are *near* (i.e. it falls within the range of *near*).

If the freeze operator is introduced between the *eventually* operators, metric constraints between intermediate points of the walkthrough can be expressed.

2.3.2 Metrics in Temporal Descriptions

Temporal freeze variables are used to retain the indexes of frames in which certain conditions hold and to express metric relationships among these indexes. For instance, the sequence assertion

$$(j) \models_t t_a.(\Diamond_t t_b.(\theta)) \quad \wedge \quad (|t_a - t_b| = soon) \tag{17}$$

expresses that, after the j^{th} frame of the sequence, a frame will be encountered *soon* (i.e. within a time interval which can be characterized by the *soon* qualifier) in which the assertion θ holds.

2.3.3 Metrics in Spatio Temporal Descriptions

The joint use of spatial and temporal freeze variables permits the representation of relationships and conditions where space and time are inherently tangled, such as, for instance, advancement and approaching of objects. By combination of spatial and temporal metric relationships, considerations about the relative speed of the objects are also possible.

The approaching of an object q towards an object p (see Fig.3) can be expressed through the following assertion:

$$\begin{aligned}(t_a) \models_t ((q) \models_s v_q.(q)) \;\wedge\; ((p) \models_s v_p.(p)) \;\wedge \\ \Diamond_t t_b.(\|v_q(t_a) - v_p(t_a)\| \leq \|v_q(t_b) - v_p(t_b)\|)\end{aligned} \tag{18}$$

which states that, freezed in v_q and v_p any two points belonging to p and q, there will eventually be a frame t_b in which the distance between v_q and v_p is lower than in the first frame t_a. Note that this basically states that, there will be a frame in which *any* couple of points in p and q will be nearer than they are in the first frame. The context declaration can be weakened through the use of negation connectives so as to state that the approaching condition holds not for all the couples of points but only for some of them.

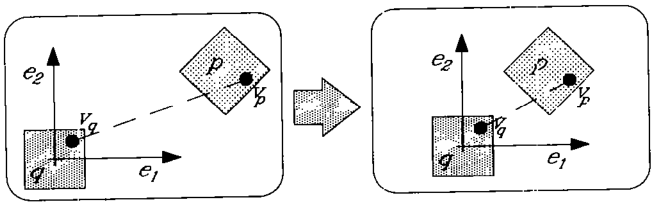


Figure 3: An object q approaching a second object p.

The expression of the advancement of an individual object requires the comparison of the positions of the same object in different frames (see Fig.4). This is accomplished by the following assertion:

$$(t_a) \models_t ((q) \models_s v_q.(q)) \ \wedge \ \Diamond_t t_b. \left((v_q(t_a)) \models_s \Diamond_{e_1^+} q \right) \tag{19}$$

which asserts that, for any point v_q belonging to q, there will be a time t_b in which the point v_q is right of the position of v_q in the initial frame.

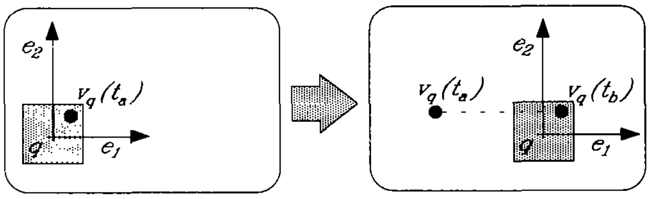


Figure 4: An object q advancing along the positive direction of its first axis.

Reasoning about the *speed* of changes in the spatial relationships between the objects is supported by the joint use of spatial and temporal quantifiers. For instance, a distance which changes of a quantity *far* in a time interval *soon* intuitively correspond to a *fast* motion. The appropriate combination of temporal and spatial relationships also permits the description of composite spatio temporal relationships such as *zigzag*, *backward and forward* or *revolution* motions.

3 Specification of Reaction patterns

Each virtual agent is natively provided with a set of *basic actions*. For instance, a virtual car is provided with basic actions to advance, blink, turn, and so on. The behavior pattern of the agent is defined by the *reaction logic* which supervises the execution of basic actions in response to different combinations of stimuli received from the environment. For instance, referring again to the example of the car, the reaction logic could be defined so as to ensure that the car stops when a second car is approaching from its right hand side or when a pedestrian-like agent is encountered on a zebra-crossing.

Following the operational approach to the specification of embedded systems, reaction logics are defined through a process-oriented language which supports the automatic operation of behavioral models during the execution of the virtual environment.

Briefly described, the reaction logic of an agent is made up of a set of *execution modes* and a *scheduling logic* controlling their activation. Each mode is a sequence S of basic actions, possibly including loops, random selections, and blocks:

$$S = t \mid S_1 \rightarrow S_2 \mid S^* \mid S_1 \sqcap S_2 \mid stop \tag{20}$$

The simplest possible execution mode for an agent is made up of a single basic action t. More complex modes can be constructed through the concatenation operator $\rightarrow$ which permits to chain the execution of multiple sequences, or with the loop operator * which permits the indefinite repetition of a sequence. The random selection operator $\sqcap$ allows for the specification of non-deterministic choices taken by the agent. Finally, the *stop* action defines a blocking condition in which the mode does not engage the execution of any further action.

The operation pattern of the agent results from the interleaved execution of the sequences of its modes. This operation occurs under the supervision of the *scheduling logic* which activates and disactivates execution modes in reaction to the occurrence of external stimuli. When a mode is *activated*, its sequence is executed starting from the first action until the mode is *disactivated* by the *scheduling logic*. Multiple modes may be concurrently active.

The scheduling logic of an agent is suitably described by a *condition-event* Petri net model [12]:

$$Scheduling\ Logic = < C, E, A > \tag{21}$$

where:

- C is a set of *conditions*, each associated with a mode or with an external stimulus. Conditions associated with modes, represented as double circles, are either *true* or *false* whether their corresponding modes are active or not. Conditions associated with stimuli, represented as circles, are true when their corresponding stimuli are being perceived and false otherwise. True conditions are marked with a black token to distinguish them from false conditions.

- E is a set of *events*;

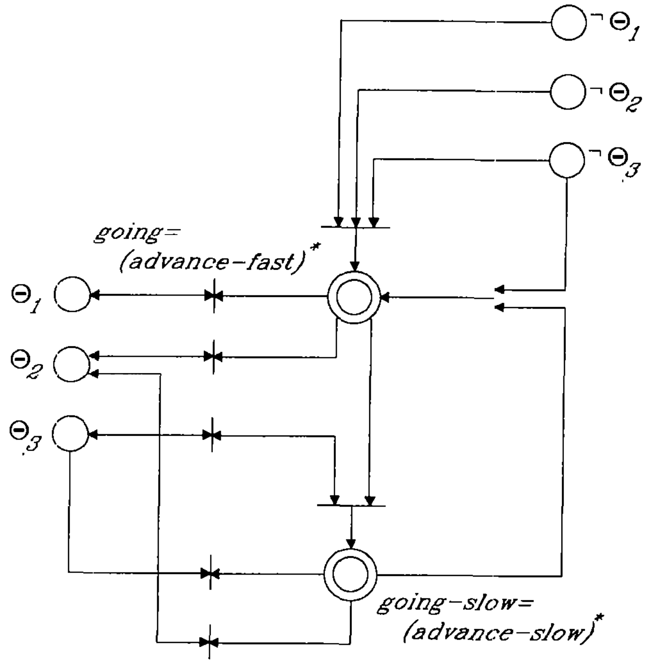
- A is a set of pre-condition and post-condition arcs associating conditions with events or viceversa. In the graph, pre-conditions and post-conditions are represented as arcs directed from conditions to events or viceversa.

The status of truth or falseness of conditions change over time according to the execution rule of events which is defined in the following three clauses:

- *Firability:* An event is *firable* if all the conditions that are connected to it through a pre-condition are true;

- *Weak Fairness:* If an event is firable then it will eventually fire or get unfirable;

- *Firing:* When event e is fired: (1) all the conditions that are connected to e through a pre-condition or through a post-condition are removed from the set of true conditions, (2) all the conditions that are connected to e through a post-condition are inserted in the set of true conditions.



Θ_1 = *car appproaching from right at near or medium distance*

Θ_2 = *pedestrian in front at medium distance*

Θ_3 = *pedestrian in front at near distance*

Figure 5: An operational model for a simple behavior pattern of a car-like agent.

As an example, in Fig.5, a simple model is presented which captures the behavior pattern of a car-like agent. The car is associated with two modes, referred to as *going* and *going-slow*, which consist in the cyclic execution of a fast or slow advancement action. The agent is sensitive to six different stimuli, corresponding the the presence or absence of a pedestrian-like agent in front at a distance near or medium, and to the presence or absence of a car approaching from the right hand side. If no cars are approaching from the right hand side and no pedestrian-like agents are on the road at a distance medium or near, the *going* mode is scheduled and the car advances at fast speed. In this condition, if a pedestrian is found at a distance medium along the road, the *going* mode is disactivated and the mode *going slow* activated, and the car advances with low speed until no pedestrian agents are in the range of medium distance and the *going* mode is entered again. Besides, whatever mode is executing, if a pedestrian is found at a distance in the range of near or if a car approaching from right is detected, both the modes are suspended so that the car stops until the *going* mode is entered again.

The operational model which defines the reaction logic of an agent permits a natural translation into code. In this translation, each agent is associated with an *automaton* which operates its condition-event net and interleaves calls to the active modes so as to execute them step-by-step. Besides, a *global parser* observes the virtual scene to detect the occurrence of spatio-temporal evolutions corresponding to stimuli that are relevant for any of the virtual agent. Whenever one such stimulus is detected, the parser sets true the appropriate condition in the scheduling logic of the interested agent thus conditioning the execution of its operation automaton. It is worth noting that, in this type of implementation, the automata associated with agents basically behave as coroutines: on each activation they are able to execute a number of steps in the operation of their corresponding agent and then release control until the next call. This allows for the implementation of light weight scheduling policies which permit to run a large number of concurrent agents (each executing a number of concurrent modes) within a single thread of control, thus relieving the operating system from the overhead deriving from multi-thread control.

4 A Tool for the Interactive Training of Virtual Agents

Specifications of reactive behaviors according to XSTL spatio-temporal assertions and Petri-net based operational models may result in very burdensome coding even for simple dynamic virtual scenes. This hurdle may be overcome by the use of appropriate visual interfaces supporting visual specifications by example of agent behaviors, provided that automatic parsing of visual specifications and code generation of behavior models are available.

To this end, a prototype system has been developed, based on the representation techniques expounded in the previous sections, for the training by example of dynamic scenes in virtual environments. In this system, the user may depict a 3D virtual scene, and train one specific agent to assume selected behaviors in reaction to the occurrence of spatio-temporal relationships with other agents populating the scene. The system runs on a IBM RISC 6000 machine with 7234 GTO for real time interaction support with the 3D virtual world. This is visualized through Gra-phigs under X-windows. 3D objects and 3D scenes are graphically constructed using a separate software package (3D-studio) and downloaded into the system environment. Here, motion patterns are defined for the objects that must exhibit some dynamics in order to define specific dynamic situations to which the trainee has to react. A single agent may be trained for different reactive behaviors to multiple situations in order to build even complex "reaction patterns".

Reaction pattern specification is carried out visually, by example, separately for each agent: in the training session, the user identifies himself in the object to be trained and uses special buttons to activate the basic actions of the agent itself at the occurrence of some spatio-temporal relationships with the other objects. Reactive behaviors so specified are encoded according to the operational model and added to the object description.

In Figs.6 and 7, a case example of designing a dynamic virtual environment for giving driving lessons to a beginner is presented. In this environment, the beginner must drive a car virtually in a city and interacts with the other agents in the virtual world. Rules of "good driving" are supposed to be learnt by the beginner during his driving, by observing the behaviors of the other agents and the effects of his actions in different situations.

Agents in the environment are limited to be pedestrians and cars for the sake of simplicity. Cars are trained with right driving rules with respect to pedestrians and other cars, according to the behavioral model given in Sect. 3.

Fig.6a depicts the virtual scene that has been created for the definition of car behaviors. The designer drives the virtual car, and a pedestrian is made crossing the road when the car is at a *medium* distance (see Fig.6b). Acting on the buttons, the designer lowers the speed and hence stops when he get near to the pedestrian (see Fig.6c). After, he restarts. Spatio-temporal relationships between virtual objects are detected by a parser which translates these relationships into XSTL clauses. These are associated with the activations of the basic actions of the car (lower the speed, stop, restart) in a simple behavioral net. If training is satisfactory (the trained behavior can be played back for checking) the behavioral net is added to the car model.

In Fig.7, the training of the behavior with respect to other cars in the proximity of a crossroad is shown. Here again the designer drives the virtual car (the one on the left of the screen) and gets closer to the crossroad lowering the car speed (Figs.7a). As a consequence, the spatio-temporal clause "approaching to a crossroad being at a medium distance" is associated to the basic car action "lower the speed". As a second car appears on his right side at "medium distance", he stops his car (Figs.7b and 7c). As an effect, the spatio-temporal clause "a car is approaching from the right side" is associated with the disactivation of the advancement condition. These condition-reaction associations are again added to the car model if satisfactory. After he restarts. Behavioral nets with multiple behaviors have a comb-

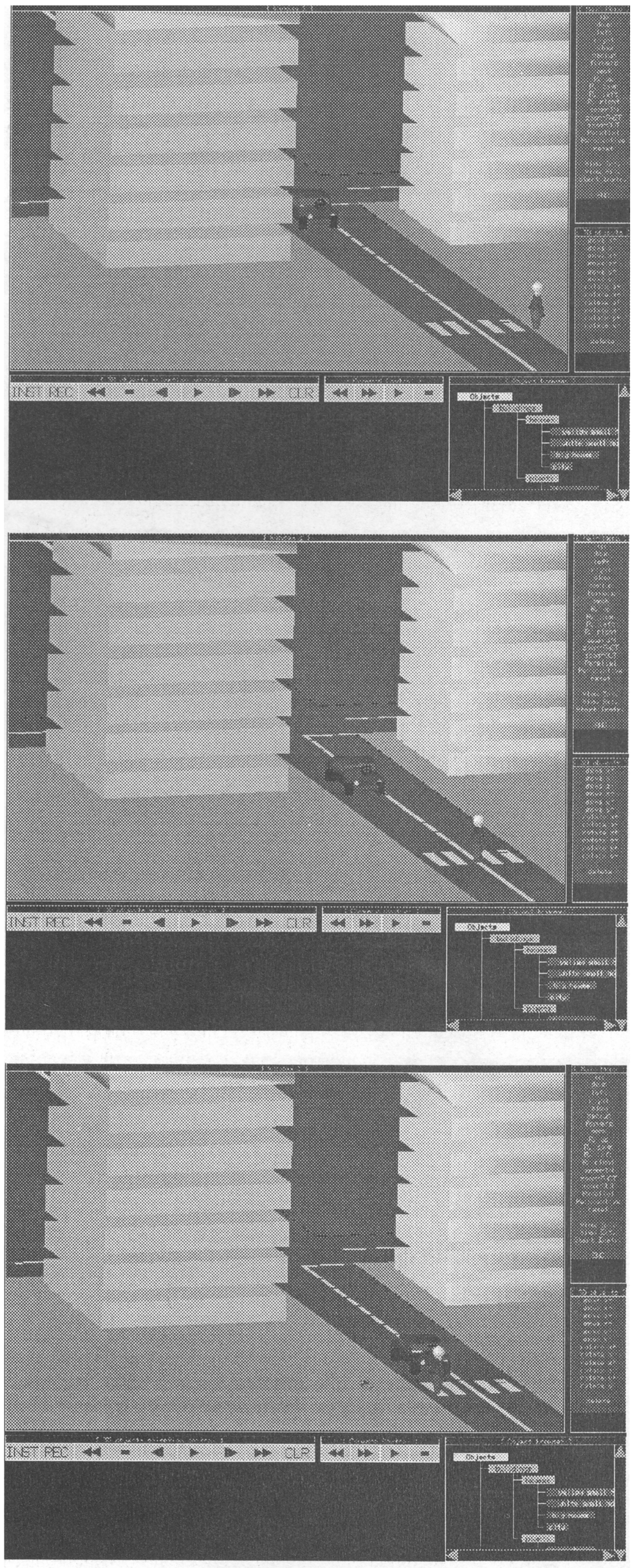


Figure 6: Visual programming of a car agent: behavior in the presence of a pedestrian.

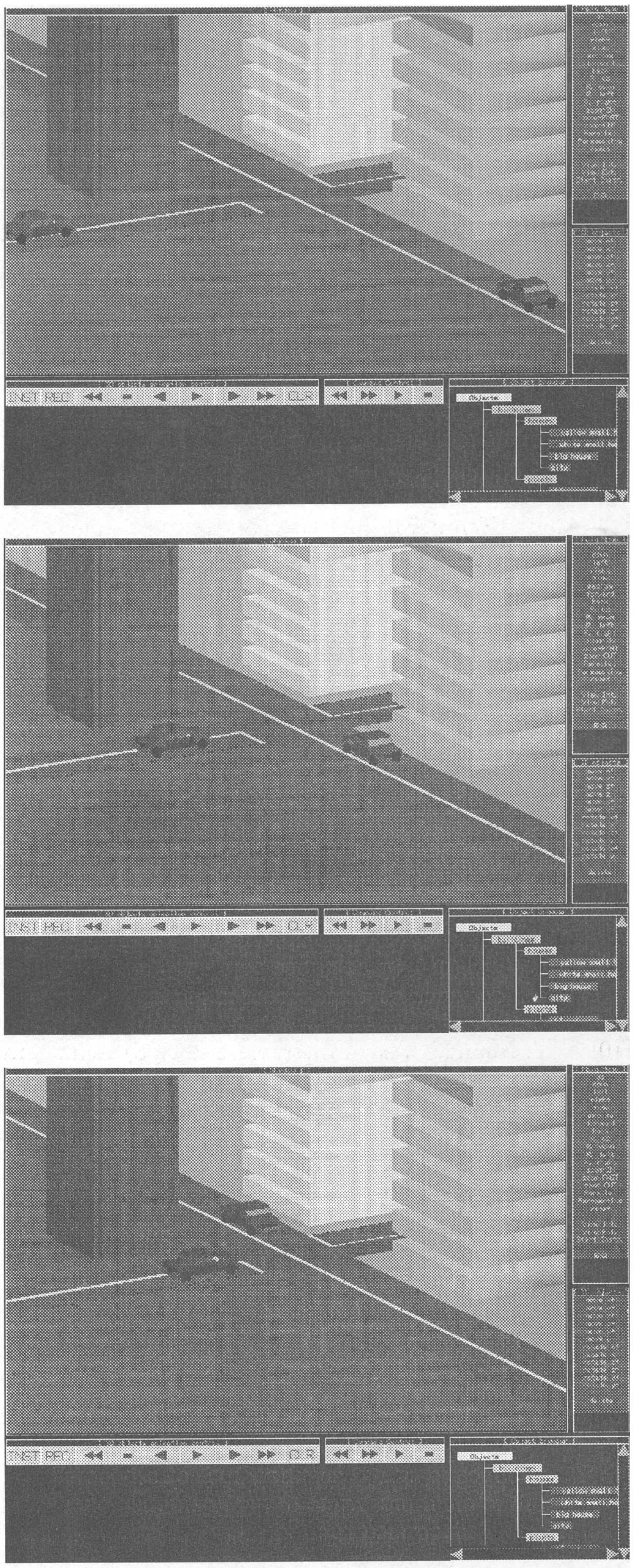


Figure 7: Visual programming of a car agent: behavior in the presence of a car at a crossroad.

like structure that makes easy their maintenance and changes. If a new behavior is specified for the same object, this is simply added as a comb tooth. Changes to behaviors that have been specified result into additions/deletions to the appropriate subnet, depending on the type of change-mode selected. If "change with additions" is selected, for every spatio-temporal pattern detected by the parser, subnets that include this pattern as a precondition are activated; as a new action pattern is specified this is simply added to the existing subnet(s) leaving unchanged the rest of the subnet. If a "change with deletions" is selected, also in this case for every spatio-temporal pattern detected by the parser, subnets that include this pattern as a precondition are activated; but differently from the previous case, as a difference in the condition-reaction patters is detected between the previous and the actual definition, the old definition is deleted and replaced by the new one.
In Fig.8, some situations are shown, concerning the interaction of a driving beginner with the virtual world previously designed, is shown. Here the beginner is driving the car on the right side of the screen. The other car in the virtual environment is expected to exhibit the behaviors trained in the previous stage. In the first case, the beginner driver car reaches first the crossroad and the cars coming on his left side is expected to give it the right of way according to the training (see Fig.8a and 8b). After the car has passed, the other restarts. Since a pedestrian is on zebra-crossing in the car lane, it stops again to make the pedestrian to pass over. (Figs.8c and 8d).

5 Conclusions

In this paper, visual specification by example of behaviors of virtual agents has been addressed. Visual specification by example is believed to largely reduce efforts of coding and to make easy maintenance of dynamic virtual worlds. Agents are trained by the designer by visually replicating reaction patterns to specific spatio-temporal relationships with other agents in the virtual world. Visual specification is formally supported by an appropriate language for the qualitative representation of spatial relationships between objects, as well as by an operational model of behaviors.

References

[1] R.Alur, T.Henzinger, "A really temporal logic," *Tech.Rep. 92-1310 Dept. of Computer Science, Cornell University, Ithaca, New York*, Nov.1992.

[2] S.K.Chang, Q.Y.Shi, C.W.Yan, "Iconic Indexing by

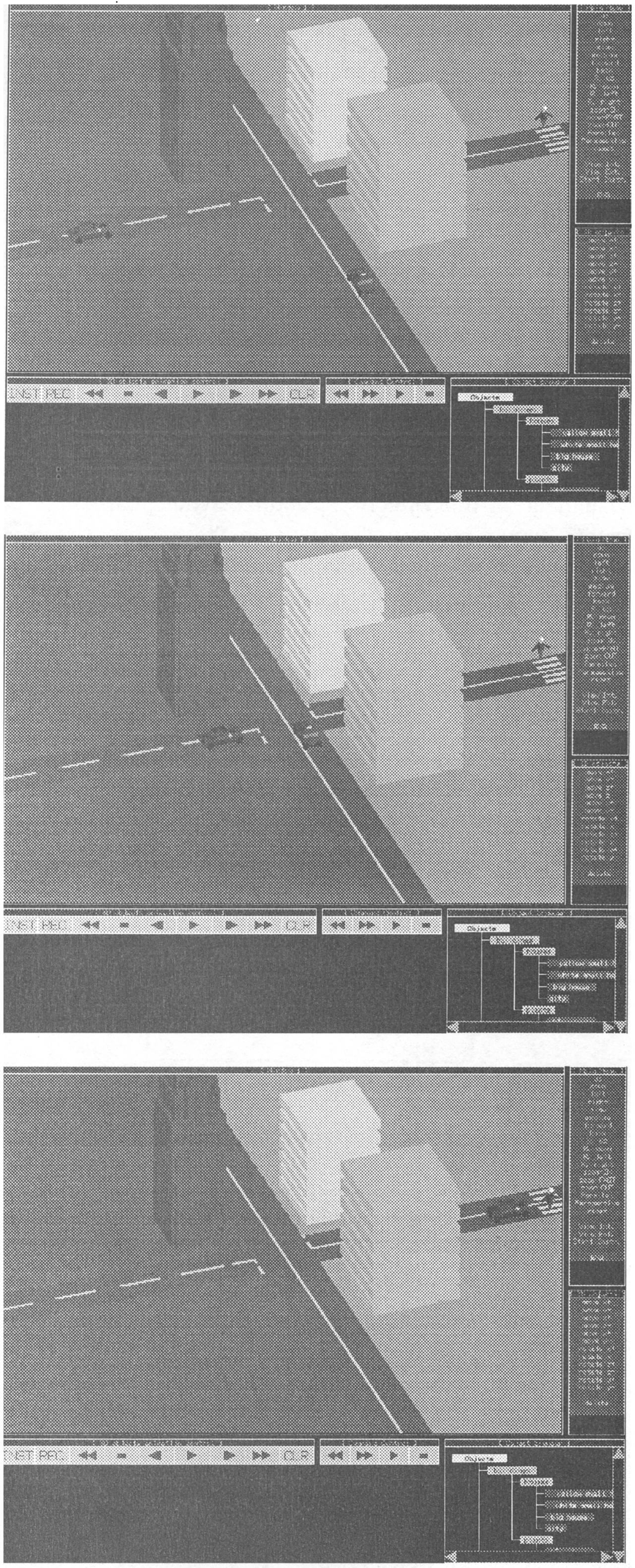


Figure 8: Interaction with the virtual environment with trained agents.

2-D Strings", *IEEE Tr.on PAMI*,Vol.9,No.3,July 1987.

[3] A.Del Bimbo, E.Vicario, D.Zingoni, "Sequence Retrieval by Contents through Spatio temporal Indexing," *Proc. IEEE VL'93 Workshop on Visual Languages*, Aug.1993, Bergen, Norway, 1993.

[4] A.Del Bimbo, E.Vicario, D.Zingoni, "A Spatial Logic for Symbolic Description of Image Contents," accepted for publication on *Journal on Visual Languages and Computing.*

[5] A.DelBimbo, E.Vicario, D.Zingoni, "Design of Virtual Worlds," *Proc. VL'93 IEEE Symposium on Visual Languages*, Bergen NW, 1993.

[6] A.Del Bimbo, E.Vicario "A Logical Framework for Spatio-Temporal Indexing of Image Sequences," in Spatial Reasoning, S.K.Chang, E.Jungert, and G.Tortora (Eds.), to be published by Plenum Press.

[7] A. Frank, "Qualitative Spatial Reasoning about Distances and Directions in Geographic Space", *Journal of Visual Languages and Computing*, vol. 3, pp. 343-371, 1992.

[8] J.B. Lewis, L. Koved, D.T. Ling, "Dialogue Structures for Virtual Worlds", *Proceedings of CHI'91*, New Orleans, 1991

[9] S.K. Helsel, J.P. Roth, "Virtual Reality, Theory, Practice and Promise", Meckler Ed., Westport London, 1991

[10] K.T. Huang, "Visual Interface Design Systems", in "Principles of Visual Programming Systems", S.K. Chang Ed., Prentice Hall Ed. Englewood Cliffs, NJ, 1990

[11] Z.Manna,A.Pnueli,"The Temporal Logic of Reactive and Concurrent Systems," Springer Verlag, New York, 1992.

[12] T.Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, Vol. 77, No. 4, Apr. 1989.

[13] A.J. West, et al., "Aviary - a Generic Virtual Reality Interface for Real Applications", *Tech.Rep.1992 AIG Dept. Computer Science, Univ. of Manchester, Manchester, UK*, 1992.

[14] "Virtual Reality Distributed Environment and Construction Kit (VR-DECK): User's guide," Virtual Worlds Group, IBM T.J.Watson Res.Center, Yorktown Heights, NY. May 1993.