Integrating miniMin-HSP agents in a dynamic simulation framework

Miguel Lozano[1], Francisco Grimaldo[2], Fernando Barber[1]

[1] Computer Science Department, University of Valencia,
Dr.Moliner 50, (Burjassot) Valencia, Spain
[2] Institute of Robotics, University of Valencia,
Pol. de la Coma s/n (Paterna) Valencia, Spain
{Miguel.Lozano}@uv.es

Abstract. In this paper, we describe the framework created for implementing AI-based animations for artificial actors in the context of IVE (Intelligent Virtual Environments). The minMin-HSP (Heuristic Search Planner) planner presented in [12] has been updated to deal with 3D dynamic simulation environments, using the sensory/actuator system fully implemented in UnrealTM and presented in [10]. Here, we show how we have integrated these systems to handle the necessary balance between the reactive and deliberative skills for 3D Intelligent Virtual Agents (3DIVAs). We have carried out experiments in a multi-agent 3D blocks world, where 3DIVAs will have to interleave sensing, planning and execution to be able to adapt to the enviromental changes without forgetting their goals. Finally, we discuss how the HSP agents created are adequated to animate the intelligent behaviour of 3D simulation actors .

1 Introduction and previous work

Artificial humans and other kinds of 3D intelligent virtual agents (IVA) normally display their intelligence through their navigation skills, full-body control, and decision-taking formalisms adopted. The complexity involved in these agents, normally suggests designing and executing them independently of the 3D graphics engine, so the agent could be focussed on their behavioural problems (see figure 1).

There are numerous applications that would require these kinds of agents, especially in fields such us, entertainment, education or simulation [2]. We are working towards the creation of a robust simulation framework for IVE simulations, where different 3D embodied agents are able to sense their environment, to take decisions according to their visible states, and finally to navigate in a dynamic scenario performing the actions which will animate their behaviours in real time.

There has been a great amount of research along the main behavioural requirements of 3DIVA systems, from their physical appearance and motor system to the cognitive one, as described in the spectrum introduced by Ayleth in [1].

Early work in AI planning archiectures for 3DIVAs was undertaken by Badler et al. They proposed an architecture based mainly on two components: a) Parallel state-machines (PaT-Nets), which are good at sequencing actions to support Jack's high level behaviours, and b) the low level (reactive) loop (sense-control-act, SCA) used to handle low level information mainly used for locomotion. [2]. SodaJack [5] and Hide-and-Seek [3] systems are created through a combination of two planners: a hierarchical planner (ItPlans [4]), which controls the main agent behaviours, and a specific purpose search planner, devoted to help Jack in locating objects or other agents. In SodaJack, the character interleaves planning and execution in a single-agent simulation, where reactivity and interaction wasn't considered. On the other hand, Hide-and-Seek simulations introduce a multi-agent framework in a dynamic environment, although the planning schema remains the same. In this case, agent reactivity is achieved by comparision between the perceived world state and the partial hierarchical plan, that is regularly revised [3].

Behavioural animation based on Situation Calculus is another cognitive approach which has been adapted to virtual creatures [9]. However, the state space model designed is more close to a declarative language than an agent centered behavioural system.

Interacting Storytelling systems integrate AI techniques such as planning with narrative representations to generate stories [14]. In [13], we discuss the use of HTN's [6] and HSP planning formalisms in Interactive Storytelling from the perspective of story generation and authoring. The main difference between these systems and the one presented here lies in the introduction of the perception system presented in [10] which let the agents deal with partially observable 3D dynamic environments.

Beliefs-Desires-Intentions (BDI) has adopted a significant number of implemetations in order to build cognitive agent architectures. A reduced number of them has also been applied to the current context, as VITAL [7] and SimHuman [8] platforms have shown. SimHuman shows a real-time platform for 3D agents with planning capabilities, which represents from a 3DIVE perspective, a similar approach to the multi-agent system presented in this paper. However, we will concentrate on the planning formalism introduced and also how agents deal with reactivity in the behavioural system designed.

Accordingly to this, the aim of this paper is to present a new agent system which interleaves sense, plan, and execution tasks to deal with normal 3DIVE's multi-agent simulations, where normally intelligent characters modify and interact autonomously with the 3D environment, guided by their AI based formalisms.

The next section shows an overview of the 3D multi-agent architecture implemented, where the world modelling and sensory system fully integrated in UnrealTM are briefly explained. Section 3 is focussed on the behavioural control of the simulation agents created. We analyse the planning algorithm for dynamic environments and the behavioural system designed to handle reactivity and goal direction for 3DIVAs. Finally, section 4 shows the results obtained from this framework in a *shared* blocks world, where several agents have been introduced to create an intelligent simulation environment.

2 System Architecture Overview

The multi-agent system designed is based on a distributed model that figure 1 shows. As mentioned before, this modular structure is adequate for 3D real time graphic simulations, as it provides for a scalable and reusable framework for both, 3D agents and environments.

As figure 1 shows, the architecture implemented is mainly divided into two components: the world manager, responsible for managing the information flow between the agent and the 3D environment (sense/act) [10], and the behavioural system of the agent, devoted to handling reactivity and planning for the 3DIVAs created (possibly running on a separate process or machine).

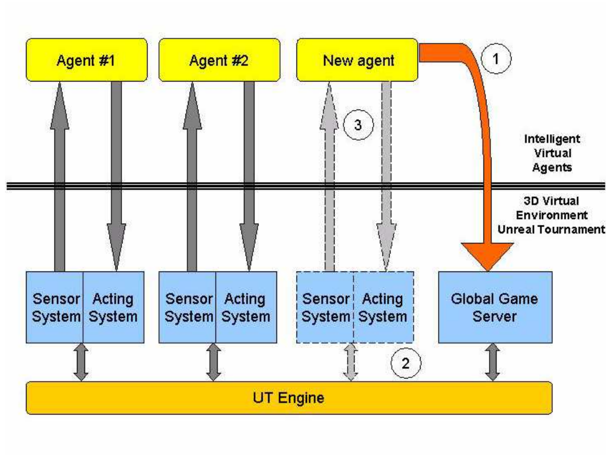


Fig. 1. System architecture overview

2.1 Behavioural Agent System

This system is the responsible for controlling the main components of the task-oriented agent architecture that figure 2 shows. The main components are briefly described now:

- **The Agent Control Module** contains two important agent components: a) the Agent Memory (perceived world and internal state) and b) the Task Control Model.

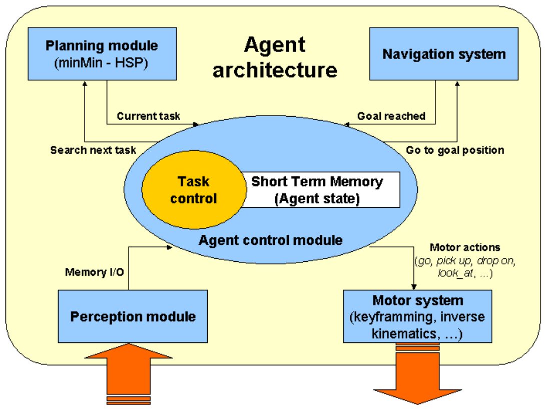


Fig. 2. Internal agent architecture

The Agent Memory is a dynamic object container, which asynchronously receives and updates new information from the perception module. We mantain two representation levels: low level information (location, size, ...) and symbolic object centered data (properties and their perceived values).

The Task Controller governs the agent activity anytime and it decides what to do depending on the agent and world states. Figure 3 shows the Finite State Machine (FSM) designed for general simulation task monitorization. We are using a classical task definition, so tasks consist of several primitive actions to be executed by the agent sequentially. IDLE will be the initial and the final state, where the agent has nothing to do. Anytime the agent generates a plan it goes to the WORKING state, and start sequencing the current task in its corresponding actions. WORKING is designed to transite to NAVIGATE or EXECUTE depending on the current action to carry out. NAVIGATE is simply used to undertake the *go_to* action, and EXECUTE will send an action request to the motor agent system and will wait until the results are known. To detect when the current task (and plan) should be aborted, the preconditions of the current task are regularly revised in these states (WORKING, NAVIGATE, EXECUTE). SLEEP is a state where the agent has no plan to carry out, normally this situation is motivated by world changes that finally hide the agent's goal states. To animate this situation the agent will look at the desired object and it will wait until a new possible plan to carry out is achieved. In order to do this, the agent will periodically translate from this state to the SEARCH one.

– **The Reactive Navigation System** of the agents created is based on the 3-layered Feed Forward Nerural Network presented in [11]. This local navigation system is allowed to access the agent memory, where visible and remembered objects are located. The main objective of this system is to guarantee NAVIGATION free of obstacles in a multi-agent environment.

– **Planning module** starts from the miniMin-HSP planner shown in [12], and it will be described in further detail in the next section.

3 Planning Module

From planning's point of view, all the agents are immersed in a highly dynamical scenario - many agents may be working in the same area at the same time, continuously transforming the environment.

A dynamic environment means that the planner must be able to deal with non deterministic actions. However, it must be noted that the non determinism may be of two different natures. The first is that the action by itself may have different results with different probabilities for each result, as for example throwing a dice, which has six different possible results with probabilities 1/6. We will call these actions pure non deterministic actions. The second kind of nondeterministic action is an action that, in an ideal world with no interferences is deterministic (for example, to *pickup* a block in a blocks world planning problem), but in a real multi agent world, this action may fail due to a change in the world that doesn't allow the action to be finished (other agent got the block), or the action may succeed but the resulting state is a state that comes from a composition of different actions, that are casually executed at a similar time by different agents . We will call these actions casual non deterministic actions.

The way to deal with pure non deterministic actions is to model the problem as a Markov Decision Problem (MDP) or a Partially Observable Markov Decision Problem (POMDP) [16] where the possible states resulting from the actions have a probability, and an optimal solution is a policy that has a minimum expected cost. Algorithms that deal with these problems are the Minimax Real Time Heuristic Search [17] and the RTDP [16].

However, in a virtual environment, the most common kinds of actions are the casual non deterministic actions. These kinds of actions have one expected resulting state with high probability and many unexpected resulting states with low probability. During the planning process, we consider these actions as deterministic ones, so each agent starts planning from the current state perceived from its sensors under classical planning assumptions. In this way, planning is used to generate the necessary agent intentions.However, it is necessary to choose a robust algorithm in order to recover from *perturbations*, normally when the expected results are not achieved.

Another challenge that the planner must face is that it must work in real time. The planner is used for a visual simulation, so it shouldn't take more than a few seconds to choose the action to execute.

The technique commonly used to solve these problems is to interleave planning and plan execution. In this way, it is not necessary to obtain a complete plan from the initial state to the goal state before beginning the execution of the first action. The agent only needs to advance the plan sufficiently to choose the first action, then it can execute the action and continue the planning from

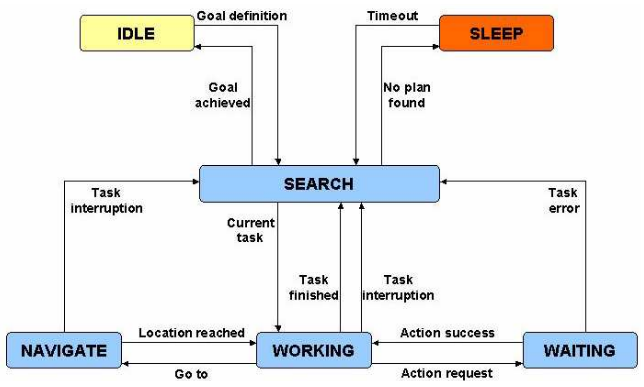


Fig. 3. Task Controler

the resulting execution state, until it is able to choose the next action, repeating the cycle until the goal state is reached.

The algorithm we use to control the interleaving of planning and action execution is a greedy search, as described in [16]. We have also included memory facilities to avoid past states. The different steps of the algorithm are represented in figure 4, where $c(a, s)$ is the cost of applying the task a in the state s, and $h(s_a)$ is an estimation of the cost from state s_a to the goal state.

1. Evaluate each task a applicable in current state s as

$$Q(a, s) = c(a, s) + h(s_a)$$

where s_a is state predicted after doing a in s.

2. Apply action a that minimizes $Q(a, s)$, breaking ties randomly
3. Observe the resulting state s'.
4. Exit if resulting state s' is a goal state, else set s to s' and go to 1.

Fig. 4. Greedy Search Algorithm

It can be clearly seen that this algorithm implements a sense - plan - act cycle. Step 1 corresponds to the planning phase, step 2 to the action execution and step 3 to the sensorization phase. The sensorization is fundamental for dealing with the non determinism of the actions, as the agent can't know the resulting state of an action. If the action observed in step 3 (s') is not the same as the predicted state (s_a) the algorithm continues the search from s'. To deal with the possible failure of the actions, we have made step 2 of the algorithm interruptible. For example, in the multi-agent blocks world scenario, agent1 may plan the task of $Picking\ up\ block1$. This task is translated into two primitive actions: $go\_to\ block1$ and $pick\_up\ block1$. But it may happen that while agent1 is going to $block1$, another agent takes it. In this case the task is no longer possible and the current action is interrupted, so the algorithm continues with step 3 for identifying the new state and planning once again.

For the planning phase of the algorithm, which corresponds with step 1 of the previous algorithm, we use the Minimin algorithm [15], which is similar to the Minimax algorithm but more oriented to single agent real time search. This algorithm searches forward from the current state to a fixed depth and applies the heuristic evaluation function to the nodes at the search frontier. We use an alpha pruning to avoid searching all the frontier nodes. This alpha pruning is similar to the alpha-beta pruning of the Minimax algorithm, and it has been shown to be very efficient with high branching factor [15]. The heuristics we are using for the Minimin algorithm consists of a relaxed search (no preconditions considered) until a goal state or the maximum depth are detected. The nodes at this maximum depth (the heuristic frontier) are given a heuristic value according to the atoms distance to the goal state.

The Planner Module we have described has the advantage of being very robust at any perturbation or unexpected change in the world and also efficient enough for the purpose of the module, although the quality of the solution (with respect to the optimal one) relies heavily on the heuristic used. However as occurs in other behavioural simulation domains such as storytelling, the optimal plan is not really a requirement. Furthermore, it is easy to realise that to extract plans in a multiagent environment it is insuficcient to guarantee that the goal state will be reached by the agent, as it is always possible to create another agent (*agent*2) that undoes the actions done by the *agent*1, independently of the algorithm used.

Other similar algorithms to the one presented here are the RTA* or LRTA* [15]. Although they need some adjustment to be able to function in a highly dynamical environment.

4 Results

The flexibilty of the simulation system created lets us design a high number of experiments in different 3D simulation environments. Furthermore, as occurs in storytelling, the full potential of story generation derives from the interaction of character behaviours while they compete for action resources (pre-conditions). As occurs in storytelling, in this case, the story can only carry forward if the character has re-planning capabilities.

Figure 5 shows the trace of one of the simulations performed in a BlocksWorld inspired 3D environment, composed of 4 tables, 4 cubes and two agents. The initial state perceived by both agents is the same, however their goals are independent. Agent1 has to invert the cubes0-3 which are placed on table0, and Agent2 will do it with cubes2-1, placed on table2 (tables1,3 are free). Although it is clear that an optimal plan, in terms of the total number of actions performed by all actors is possible to achieve, we are more interested now in checking the robustness of the planning system created, as it will have to face complex situations where the goal state can move away or even disappear. For example, initially the Agent2 decides to move the cube3 to the cube2, however Agent1 gets the cube2 before. This situation is detected by Agent2 who aborts its current plan and searchs again from the new state perceived, deciding this time to drop the picked cube(2) on the cube1, as it is now free.

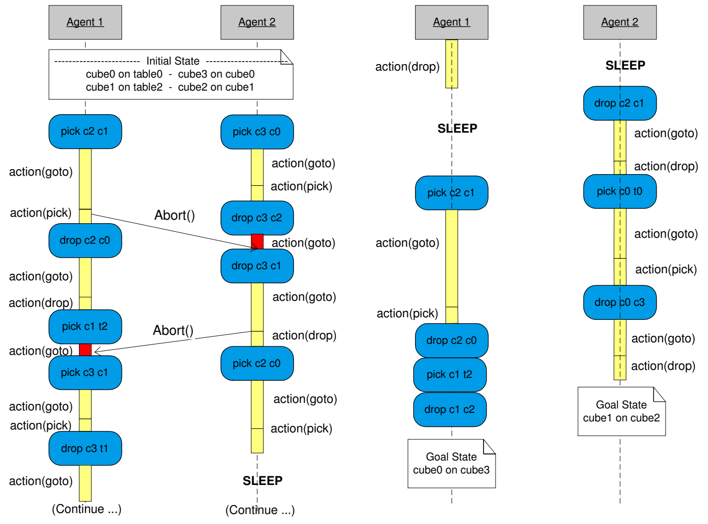


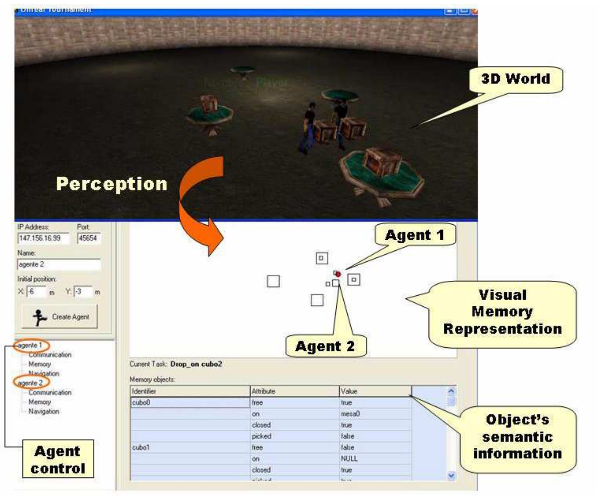
Fig. 5. Simulation trace example

As there is no muti-agent task coordination [3] between the agents, a conflict situation is generated again when both agents drop their cubes and they disturb themselves. Agent1 faces this situation, moving away cube3, while Agent2 picks up cube2. At this moment Agent2 searchs again, however, a plan can not be provided, as Agent1 is moving cube3, son Agent2 has no way to know its final location. Once Agent1 drops cube3 on one of the free tables, Agent2 decides to drop the cube previously picked on cube1 (so it will produce a new initial state from Agent1's point of view) and finally, it achieves its goals as it moves cube0 to its final position on cube3. Agent1 can finish now without more problems, so that finally all cubes are compiled in a single stack, which is a possible solution to the 2-Agent problem designed.

5 Conclusions

We have described an agent architecture that interleaves perception, planning and action, to be able to adapt itself to the changes produced by users or agents.

[3] we can consider the current agent task as its short-term intentions, while complete plans can be viewed as long-term ones



IP Address: 147.156.16.99
Port: 45654
Name: agente 2
Initial position: X: -6 m Y: -3 m

Current Task: **Drop_on cubo2**

Memory objects:

Identifier	Attribute	Value
cubo0	free	true
	on	mesa0
	closed	true
	picked	false
cubo1	free	false
	on	NULL
	closed	true

Fig. 6. Snapshot of the simulation framework in real time

We have shown how to combine planning and reactivity (based on the precondition checking performed by the agents while they are working) in order to manage complex environmental simulations in 3D. This can be also very useful for behavioural character animation, for example when a character detects that another one is moving the cube that it will need in the future, we can animate this situation through a suprise agent dialogue (eg: ... where are you going with this cube?). Reactivity tunning can be also easily introduced, as the agents can always try to follow their current task, and anly re-plan after their actions. Furthermore, it is easy to see how new informative heuristic functions can be introduced in the planning system to finally influence the agent behaviour(in a similar way as narrative concepts can guide actor's decision taking in storytelling domains). Heuristics can derive mainly from two information soruces: a) from perception: where typically object distances or other kind of situation recognition can be easily introduced, b) from the agent internal state: as showed in [13], agents could also manage some *fluents* (eg. *mood*, etc.) which finally assist its decision taking. From storytelling system's point of view, the behavioural approach presented lets the agents to autonomously deal with reactivity and long-term dependencies in similar 3D simulation scenarios (in storytelling domains, normally is the *author* who apriori introduces the narrative content using AND/OR graphs composed by independent sub-problems, so agent behaviours based on long-term dependencies can not be considered).

Summarizing, the storytelling inspired agents created are able to adapt themselves to the enviromental changes they are producing anytime, which is an important point when simulating intelligent and *believables* character's behaviours.

References

1. Aylett R. Luck M. *Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments.* Applied Artificial Intelligence, 2000.
2. N. Badler, M. Palmer, R. Bindiganavale. *Animation control for real-time virtual humans.* Communications of the ACM, 42(8). August 1999.
3. Badler, N., Webber, B., Becket, W., Geib, C., Moore, M. Pelachaud, C., Reich, B., and Stone, M., *Planning for Animation* in D. Thalmann and N. Magnanat-Thalmann (eds.), Computer Animation, New York: Prentice Hall Inc., 1995.
4. Geib, C. *The intentional planning system: Itplans.* Proceedings of the 2nd Artificial Intelligence Planning Systems Conference. 1994.
5. C. Geib and L. Levison and M. Moore, *Sodajack: An architecture for agents that search and manipulate objects.* Technical Report MS-CIS-94-16/LINC LAB 265, Department of Computer and Information Science, University of Pennsylvania, 1994.
6. Cavazza, M., Charles, F. and Mead, S.J., 2001. *AI-based Animation for Interactive Storytelling.* Proceedings of Computer Animation, IEEE Computer Society Press, Seoul, Korea, pp. 113-120.
7. George Anastassakis and Tim Ritchings and Themis Panayiotopoulos. *Multi-agent Systems as Intelligent Virtual Environments* Lecture Notes in Computer Science, Vol. 2174, Springer-Verlag, pp.381-395, 2001.
8. S. Vosinakis, T. Panayiotopoulos, *SimHuman : A Platform for real time Virtual Agents with Planning Capabilities*, IVA 2001, 3rd International Workshop on Intelligent Virtual Agents, Madrid, Spain, September 10-11, 2001.
9. John Funge, Xiaoyuan Tu, and Demetri Terzopoulos (1999). *Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters in Computer Graphics.* Volume 33 Annual Conference Series (Proceedings of SIGGRAPH 99) pages 29-38.
10. M. Lozano et al. *An Efficient Synthetic Vision System for 3D Multi-character Systems.* 4th International Workshop of Intelligent Agents (IVA03),Springer-LNAI Munchen 2003.
11. M. Lozano, F.Grimaldo, J. Molina. *Towards reactive navigation and attention skills for 3D intelligent characters.* International Work-conference on Artificial and Natural Neural Networks (IWANN). June 3-6, 2003 Mahn, Menorca (Balearic Islands, Spain).
12. M. Lozano, Mead, S.J., Cavazza, M. and Charles, F. *Search Based Planning: A Model for Character Behaviour.* Proceedings of the 3rd on Intelligent Games and Simulation, GameOn-2002, London, UK, November 2002.
13. Charles, F., Lozano, M., Mead, S.J., Bisquerra, A.F., and Cavazza, M. *Planning Formalisms and Authoring in Interactive Storytelling.* 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany, 2003.
14. Michael Young *An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment* In The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Stanford, CA, March 2001.
15. R. E. Korf. *Real-Time Heuristic Search* Artificial Intelligence 42, pp 189-211, 1990.
16. B. Bonet, H. Geffner. *Planning and Control in Artificial Intelligence: A Unifying Perspective* Applied Intelligence 14 (3), pp 237-252, 2001.
17. S. Koenig. *Minimax Real-Time Heuristic Search* Artificial Intelligence 129, pp 165-197, 2001.