Virtual Actors' Behaviour for 3D Interactive Storytelling

Marc Cavazza, Fred Charles, Steven J. Mead and Alexander I. Strachan
University of Teesside, School of Computing and Mathematics, Middlesbrough, TS1 3BA, United Kingdom
{m.o.cavazza, f.charles, steven.j.mead, alexander.i.strachan}@tees.ac.uk

Abstract
In this paper, we describe a method for implementing intelligent behaviour for artificial actors in the context of interactive storytelling. We have developed a fully implemented prototype based on the Unreal™ game engine and carried experiments with a simple sitcom-like scenario. We discuss the central role of artificial actors in interactive storytelling and how real-time generation of their behaviour participates to the creation of a dynamic storyline. We follow previous work describing the behaviour of artificial actors through AI planning formalisms, and adopt a search-based approach to planning. The set of all possible behaviours, accounting for many different instantiations of a basic plot, can be represented through an AND/OR graph. Under certain formal conditions, the solution plan can be obtained by directly searching the graph with the AO algorithm. We describe our implementation of AO* and how it addresses the specific issues of 3D interactive storytelling, such as interaction with the virtual world and user intervention.*

Keywords: Computer Games, Interactive Storytelling, Artificial Intelligence, Virtual Environments, Virtual Humans.

1. Introduction

The next generation of computer games will be increasingly based on interactive storytelling[1]. These systems will require autonomous actors with intelligent behaviour that are fully integrated in 3D graphic worlds. In this paper, we describe the development of a new technique for implementing artificial actors' behaviour within 3D graphic environments. These behavioural models are intended for use in interactive storytelling applications, i.e. next-generation computer games, where high-quality graphics will be matched by richer storyline and more sophisticated character behaviour.

There are two essential requirements for artificial actors' behaviour in interactive storytelling. Firstly, these behaviours should be fully integrated with the actor's graphical environment, in terms of both actor animation and interaction with the virtual world objects. Secondly, they should be able to take into account user intervention and generate adaptive behaviour in real-time.

To support our experiments, we have designed a simple, yet fully implemented 3D storytelling environment (using the Unreal™ game engine for graphic rendering, animation and user interaction). The interactive storytelling experiments we report in this paper are based on a simple narrative, i.e. an episode of a sitcom-like scenario. In this episode, the principal character is seeking to invite the main female character on a date. This episode will unfold as a sequence of actions that will vary according to the character's personalities and real-time user interaction.

We first introduce some fundamental concepts of interactive storytelling. In particular, we discuss the relation between actors' behaviour and dynamic plot generation. We then describe the use of planning technologies to support artificial actors' behaviour in a way that accommodates user interaction. Finally, we present a search-based approach to plan-based behaviour and show how it can integrate planning with execution of low-level actions carried in the graphic environment.

2. Interactive Storytelling and User Interaction Modalities

There are many paradigms for interactive storytelling. Bolter and Grusin[2] have suggested that in modern computer games, users tend to have a dual role as *spectator* and *director*. This probably reflects the fact that the player usually controls a specific character, indeed called "the player character", and observes the resulting action at the same time. This example is a good illustration of the trade-off between interaction ("active user") and storytelling ("passive user") that characterises computer games, often developing interaction at the expense of storytelling. In general terms, interaction emphasises user intervention and active participation, while storytelling is more about story presentation and hence is based on scene visualisation, character animation, camera movements, etc. The contradiction between interaction and storytelling has been extensively discussed by Mateas[3], Jull[4] and Young[5].

Interactive storytelling, even more than computer games, should be organised around the notion of artificial actors as main characters. These characters, as an aspect of narrative, are deeply intertwined with plot[5]. If the character can select between various actions at a given stage, the character's choice for action actually dictates the instantiation of the plot[5]. In this context, the plot can be computed dynamically from an integrated plan generating all possible behaviours for the character, depending on the specific circumstances that will result from user intervention. Under this assumption, the global storyline can be implicitly compiled in the generic plan describing all possible behaviour for the artificial actor.

Rather than playing a role himself, the user is interfering with the plot and altering the storyline as it progresses by observing the ongoing behaviours of the artificial actors. Also, in that way, he can only interfere with actions that have narrative significance within the framework created by these actions.

We can now characterise the modalities for user interaction. It should be clear that interacting with a character's plan is a higher-level task than confronting the character physically, which is still at the heart of many computer games. In interactive storytelling, as the user is essentially interfering with an agent's activity (whether to help it or to contrast it), we can describe two principal modes of interaction with the agent. The first one is to physically interfere with the set in which the agent is evolving. For instance, the user can interfere with resources required by the actor's plan, e.g. stealing a diary that the actor plans to use to acquire some information. The second one would consist in influencing the actor's "cognitive" processes, by providing it with information that would directly alter its internal state. This can be implemented through e.g. natural language instructions[6,7]. In this paper, we will mainly discuss on-stage physical intervention as the only modality of user intervention.

3. System overview and architecture:

The prototype we describe has been developed using the Unreal™ game engine. The Unreal™ environment provides most of the user interaction features required to support user intervention in the plot, such as navigating in the environment and interacting with objects in the virtual set.

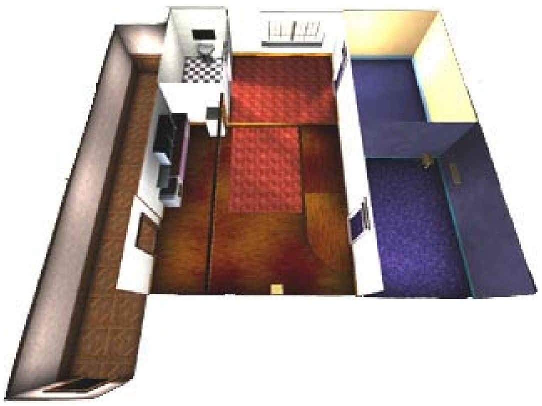


Figure 1: The Virtual set

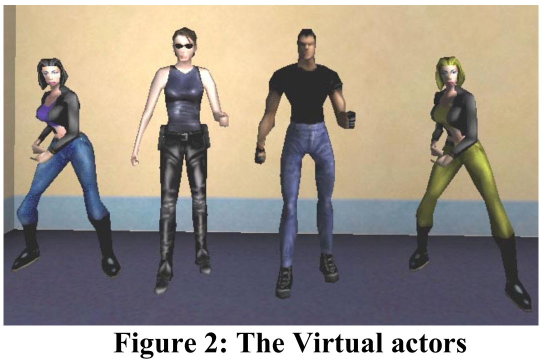


Figure 2: The Virtual actors

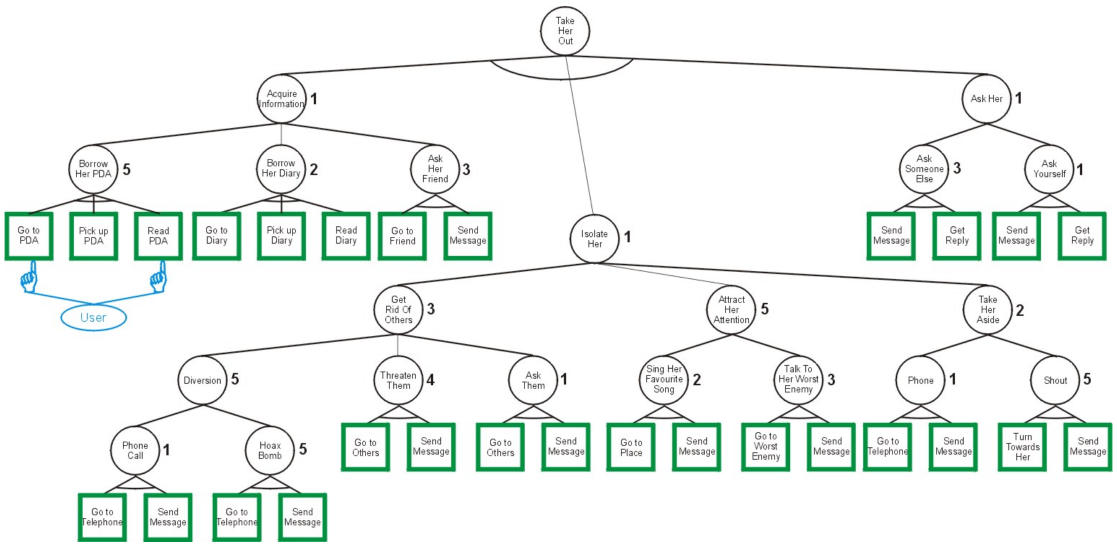


Figure 3: Unsolved "Take her out" AND/OR graph (with terminal actions)

The Unreal editor allows the creation of 3D environment layouts, the definition of various virtual human characters, as well as importing objects from various graphic formats. Besides, there exist important on-line resources for the game contents that can be tuned to the specific needs of new applications. These features, plus the high-quality of graphic rendering account for the growing popularity of Unreal™ in the development of non-gaming applications[8] or as a research tool in interactive storytelling[9] or Virtual Reality[10].

Programming in the Unreal™ environment can adopt various solutions[9]. Code redefining actors' behaviours or global control can be developed in C++ and plugged into the engine. Unreal™ also provides an integrated programming language, UnrealScript™, which gives direct access to the game API. UnrealScript™ is essentially similar in structure to Java, but is significantly slower than C++. The system described in this paper has been implemented as a set of template C++ classes, and is used as a native function by UnrealScript™. The C++ is compiled into a dynamic link library (DLL), which is loaded by the unreal engine when needed. The interface class (generated from the UnrealScript class definition by the Unreal compiler) contains the native member functions, which are called from UnrealScript, and data members that can be accessed by both UnrealScript and C++ class.

4. Plan-based Actor Behaviour

Actor behaviour in 3D environments is based on the conversion of action sequences into low-level motion corresponding to animation primitives supported by the environment's implementation[11,12]. In the case of Unreal™, low-level behaviours are represented by Finite-State Machines (FSMs). One essential requirement of these behaviours is to be integrated within the graphic environment as a whole. A good example of this is geometric path planning, where the graphic environment is first discretised into a structure on which a heuristic search procedure, such as the A* algorithm, can be applied[13]. Other important aspects of integration with the graphic environment include interaction with objects in the virtual world and user interaction.

4.1. Planning Formalism

Most of previous work in behavioural animation has focussed on low-level behaviours, such has those controlling motion and interaction with the agent's immediate physical environment. The appropriate level of description for interactive storytelling is a more abstract one, such as the one provided by planning formalisms[3,5]. Overall, there has been few real uses of planning techniques in animation, as in most cases plans tend to be compiled into scripts or execution-only finite-state automata[14,15]. The only extensive use of planning for animation has been described in the "AnimNL" project[16], in which agents are executing high-level plans using an intentional approach. This kind of representation supports the dynamic instantiation of generic plans to the specific conditions into which the action has to take place in the agent's physical environment. The intentional level is however too generic for the requirements of our application, as the actions prescribed and their expected outcome are already part of the generic storyline representation. The emphasis of our work clearly stands in the narrative representation that makes possible to use search as a planning mechanism, rather than in a more generic cognitive approach.

In this context, the search space corresponding to a plan can be described by an AND/OR graph. For instance, Figure 3 represents the example scenario we have been implementing. The top-level goal consists in inviting the female character "Rachel" ("Take her out" goal). Its sub-goals consist in acquiring information on her taste ('Acquire information'), isolating her from the rest of the group in order to be able to talk to her in private ('Isolate her') and asking her out during the conversation ('Ask her'). Each of these sub-goals is further decomposed until the planning operators correspond to terminal actions in the graphic environment (such as reading a diary, making a phone call, talking to a character, etc.). Technically, the search process that is carried out by a planning system takes an AND/OR graph and generates from it an equivalent state-space graph[17]. The process by which such a state-space graph is normally produced is called *serialisation*[18]. When the various goals are independent from one another, a planner can build a solution straightforwardly by directly searching the AND/OR graph without the need for serialising it[18].

4.2. Solution Instantiation

Since search can provide a solution[19], we have implemented a version of the AO* algorithm[20, 21, 22], which is a heuristic search algorithm producing the best solution graph from an AND/OR graph representing the problem space (there can be several solutions to the problem, i.e. several solution subgraphs). The main character will plan a solution for the scenario in order to solve the set of sub-goals and thus satisfy the main goal. The nodes in the graph are either sub-goals or terminal nodes. When the graph is created at the beginning of the scenario, each of the graph nodes is assigned a static heuristic value. The solution sub-graph produced depends on a heuristic search function, which normally is used to find the shortest/more appropriate solution. In the case of explicit graphs of moderate size (such as those representing generic character plans), this actually opens interesting perspectives for knowledge representation, as the heuristic function can be related to elements of the storyline, such as the main character's personality. The choice made on the heuristic values can be used to represent a degree of "personality" for the character. Some of the sub-goals in the graph definition do not have the same weighting depending on the consequences of their chosen actions and their overall "style". In this way, the main character can be associated a personality profile varying from shy and cautious, to outgoing and confident or sly and mischievous. The personality is part of the internal description of the character in Unreal™. Each of the nodes holds a different definition of heuristics according to the pre-defined personalities. For instance, for the "Isolate Her" sub-goal, a shy character would rather get rid of others characters using a diversion than rudely interrupting the conversation. To succeed in the same sub-goal, a sly character would certainly prefer attracting her by talking to her worst enemy to, hopefully, create the right reaction.

In section 5, we give a more detailed description of the implementation of the AO* algorithm and its use within a specific scenario. This description will emphasise the interleaving of planning and terminal actions in the graphic environment.

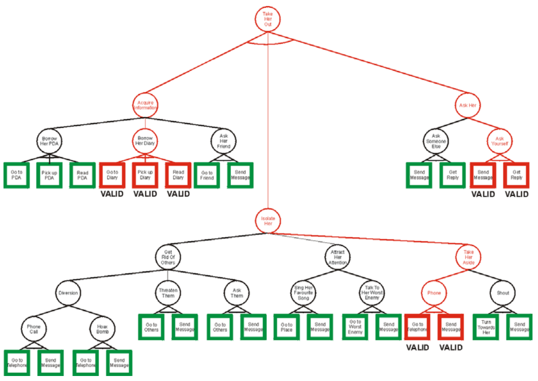
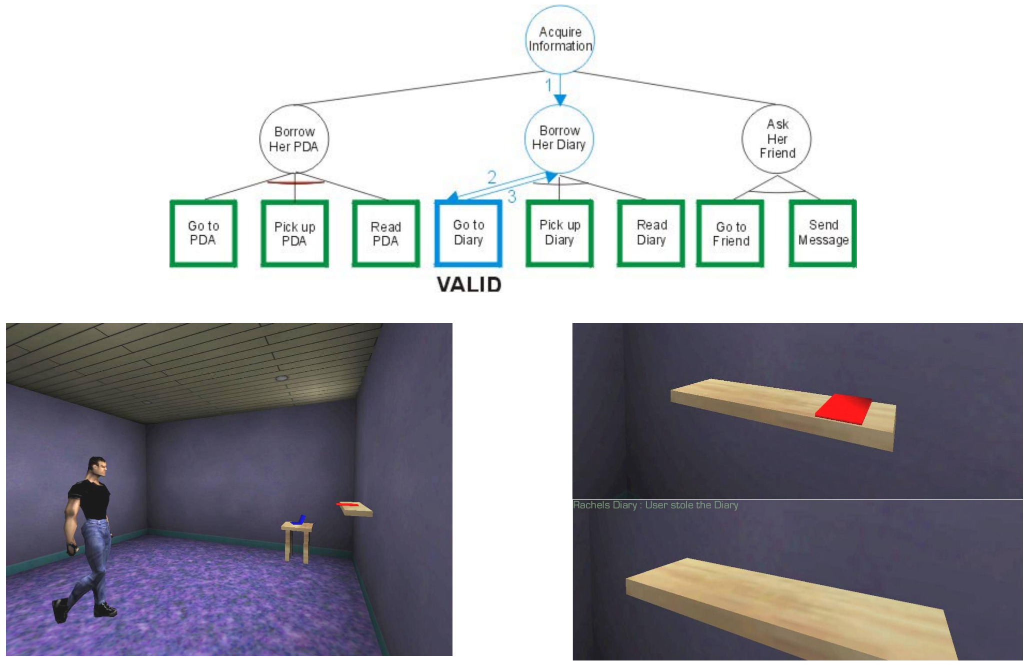


Figure 4: Example 1 solution graph



Rachels Diary : User stole the Diary

Figure 5: User intervention by stealing the diary (Example 2)

5. AND/OR Graph Search for Dynamic Behaviour

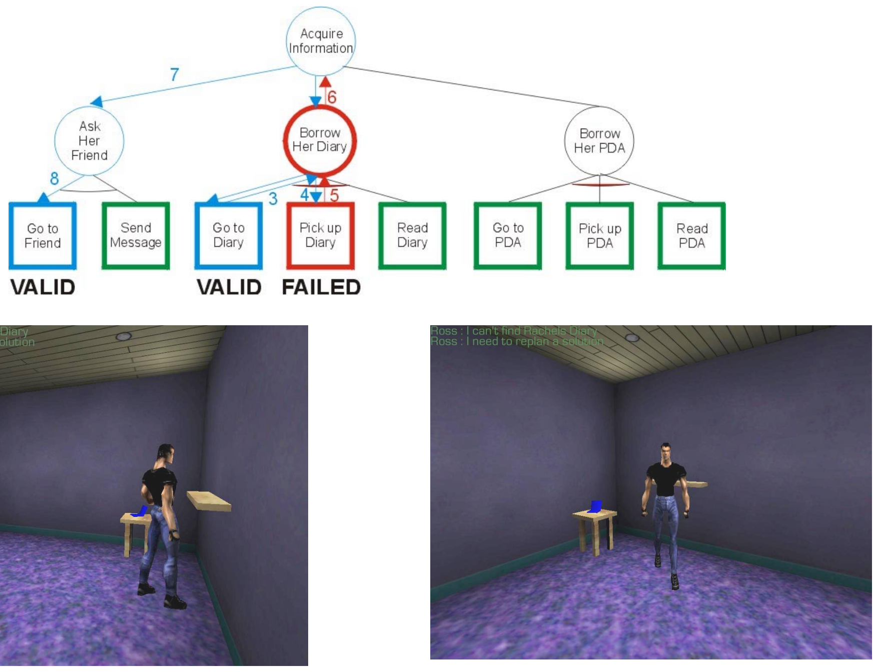
The AO* algorithm is used to find a solution graph from the AND/OR graphs which represents the global plan for the scenario, i.e. the set of all possible action sequences constituting an instance of the scenario (Figure 3).

5.1. Search Algorithm

The AO* algorithm can be described as comprising a top-down and a bottom-up component. The top-down step consists in expanding or nodes to find a solution basis, i.e. the most promising sub-graph, using a heuristic function. For instance, in the tree of figure 5 (where labelled arrows describe the graph traversal process), the "acquire information" node can be expanded into different sub-goals, such as "read Rachel's diary" or "ask a friend". The actual choice of sub-goal will depend on the heuristic value of each of these sub-goals, which contains narrative knowledge, such as the actor's personality. However, what ultimately characterises a solution graph is the set of values attached to its terminal nodes. This is why the evaluation function of each previously expanded node has to be revised according to these terminal values, using a rollback function[24], which is a recursive weighting function that aggregates individual evaluation functions along successor nodes. In the context of interactive storytelling, this bottom-up step is used to take into account action failure.

In interactive storytelling, several actors, or the user himself, might interfere with one agent's plans, causing its planned actions to fail. Hence, the story can only carry forward if the agent has re-planning capabilities. Re-planning consists in the ability to generate a new plan from updated data when the current plan becomes invalid.

One possible approach consists in re-computing the entire solution tree once the previous plan ceases to be valid: we will call this off-line re-planning. The first step consists in searching the entire AND/OR tree with AO*, and producing a complete solution sub-tree. The corresponding plan will then be executed by performing the actions attached to the terminal nodes in a left-to-right fashion that follows the implicit ordering of the tree. For each action, the associated low-level behaviour will be generated by the Unreal™ system, and the corresponding animation sequence will take place. Whenever an action fails, the heuristic value for the corresponding node is set to a "futility" value (i.e., equivalent to an infinite cost for that terminal node), and a new solution graph is computed. The new solution would take into account action failure by propagating its updated value to its



Ross : I can't find Rachels Diary
Ross : I need to replan a solution

Ross : I can't find Rachels Diary
Ross : I need to replan a solution

Figure 6: Re-planning (Example 2, cont.)

parent nodes through the rollback mechanism. In any case, failed actions cannot be undone, as they have been played on stage. Action failure is indeed part of the story itself. The computational overhead of the offline approach can become unacceptable when several actors, each acting according to their AO*-based plan, have to interact with one another.

5.2. Adding Dynamic Computation

We have thus implemented a "real-time" variant of AO* that does not compute a complete solution but rather takes advantage of the interleaving of planning and execution to only compute the partial solution tree required to carry out the next action. Like with traditional real-time search algorithms, such as RTA*[23], the solution obtained theoretically departs from optimality. The reason in our case is that the real-time variant generates the first partial solution sub-tree, whose optimality is based on the "forward" heuristic only (the rollback mechanism not being fully exploited when computing a partial solution). RTAO* finds a suitable partial solution path (though retrospectively is may not be the most optimal) through the graph. The search is a two-phase process: a forward 'explorative' phase, and a rollback phase. These actually describe how the planning and execution is performed. The forward exploration (or planning) phase will find the most suitable path by the locally optimal selection from the possible descendants. This terminates when a leaf node containing a terminal action is reached. The actor is then instructed to execute the action (execution), and the result of the execution is then propagated (via rollback) through the selected path. Upon action failure, re-planning will resume the 'forward-explorative' phase at a yet unexplored descendant (if available) within the current partial solution; conversely, success will allow the search to progress onto the development of another partial solution. This process will continue until either: all possible solutions have been exhausted and the overall plan fails, or the actor has successfully executed a series of terminal actions that constitute a successful plan instantiation.

However, the notion of optimality has to be considered in the light of the specific nature of the heuristic functions we have described, which represent narrative concepts (e.g., associated with an actor's personality, etc.). Departing from optimality in this case does not result in a "poor" solution, but rather in just another story variant (and action failure at the level of terminal nodes cannot be anticipated at the search level). Further, working on explicit AND/OR trees makes possible to design accurate heuristics! The real-time version is significantly more efficient than the off-line version in terms of CPU resources, but most

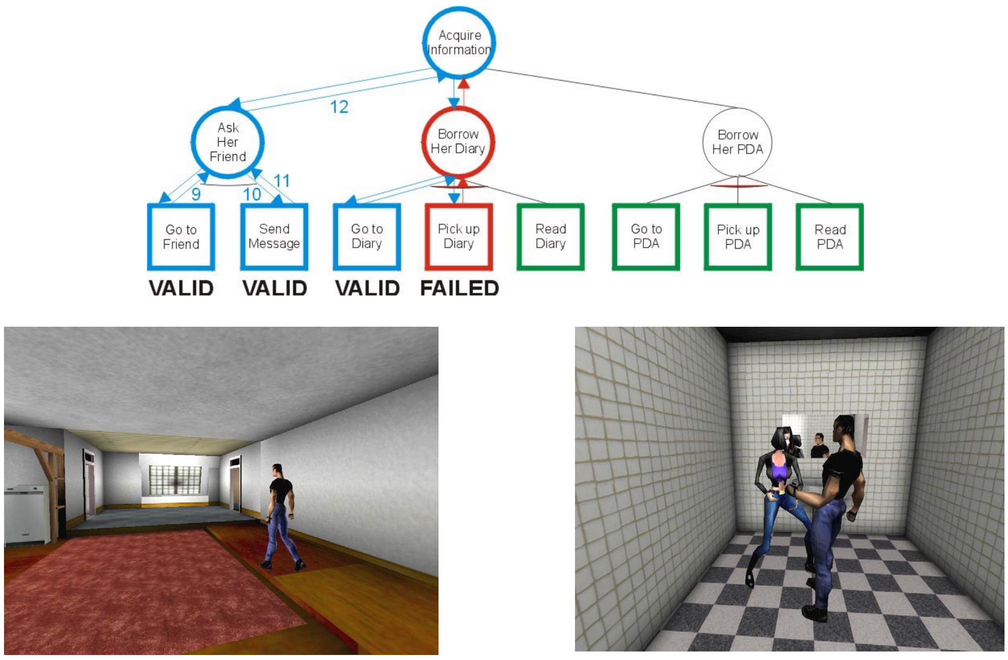


Figure 7: Partial solution for the sub-plan considered (Example 2, cont.)

importantly in terms of memory requirements. However, improvement in performance is only relevant for trees of depth greater than eight. Behaviour trees in animation have been described with an average depth of seven, in a cognitive approach[24]: however, representing even a moderately complex storyline is likely to involve much larger trees, which justifies our real-time approach.

6. Example Results

As planning is interleaved with execution, the story is incrementally constructed as a sequence of actions. Each terminal action (e.g. meeting someone) is broken down into a set of defined primitive actions that the character has to undertake. A finite state machine represents the behaviour of Unreal™ characters. Animation sequences are provided for each of the characters within the character package. The mesh animations are looped through when needed, e.g. walking, running, waiting, waving, looking around, etc. The choice of these animation sequences participates to the "dramatisation" of the character's action, which is required for the user, watching the action unfolding, to realise the narrative meaning of such actions, and possibly interfere with them in return.

Each of the terminal actions has to succeed, meaning having performed with a successful outcome, in order to complete the sub-goal. Let us consider those actions that make use of "narrative" objects (e.g., phone, diary). In the first instance, the character must reach the location where the object is supposed to be. It needs to devise a clear path to that object, avoiding obstacles: this is done by using Unreal™'s built-in mechanisms. When the object's position is reached, the precondition of "existence" has to be satisfied in order for the terminal action to be successful (in other words, this precondition is the main executability condition for the action). The partial solution is developed without considering whether its terminal actions are solvable. Thus, if the object is not available to the character, the precondition is not satisfied and the computed solution for the sub-graph is not valid any more. The relevant node in the graph representing the scenario must be updated and a new solution re-computed.

Figure 4 (Example 1) shows a solution graph where there is no user interference, thus the top-level goal ("Take Her Out") is achieved without need to re-plan. In this case, the specific action sequence only depends on the heuristic functions that reflect narrative elements (see above). Ross chooses to acquire information by reading Rachel's diary, then phones Rachel, meeting her in the main lobby and finally asks her out. As in our current prototype, user intervention takes place through interaction with the set objects, his interventions often interfere with the executability conditions[25] of terminal actions. For instance, Figures 5, 6 and 7 (Example 2) illustrate how the user can interfere with the character plan by stealing a relevant object. If the character is going to acquire information on Rachel by reading the diary (graph label 2 in graph, Figure 5), the user can contrast that plan by stealing the diary. This impairs the execution of the 'Read diary' action (graph label 3, Figure 6), *after* the character has moved to the normal diary location. The fact that the diary is missing is also dramatised, as evidenced on Figure 6. As the action fails (graph labels 3 and 4, Figure 6), the search process is resumed to produce an alternative solution for the 'acquire info' node (graph label 7, Figure 6), which is to ask one of Rachel's friends for such information. The Ross bot will thus walk to another area of the set to meet "Phoebe" (Figure 7).

7. Conclusion

The implementation of intelligent behaviour for artificial actors in 3D graphic worlds is faced with complex technical problems, such as interleaving planning and action, supporting user interaction and representing storytelling concepts.

In this context, we claim that search-based planning provides a practical short-term solution in interactive storytelling and computer games. To some extent, we would see AO*-based behavioural models as the equivalent of A*-based search for path planning, which is now almost a standard solution for computer games applications[27]. Both algorithms being a compromise between expressivity and performance. Further, the use of AO* opens additional perspectives in terms of interaction and gameplay, as its use has also been described for adversarial search in two-player games (where it shares some properties of SSS*[28]), which would provide a basis for interference with an actor's plan, either from the user or other autonomous actors.

The next step for our experiments will consist in develop large-scale AND/OR graphs for multiple virtual actors and experiment with multiple interactions between actors, which will make the consequences of user intervention more complex.

Acknowledgements

Eric Jacopin is thanked for his expert advice on AI planning issues: any remaining errors are, however, the authors' sole responsibility.

References

1. R. Nakatsu and N. Tosa. Interactive Movies, In: B. Furht (Ed), *Handbook of Internet and Multimedia – Systems and applications*, CRC Press and IEEE Press, 1999.
2. J. D. Bolter and R. Grusin. *Remediation: Understanding New Media*. Cambridge (Massachusetts), MIT Press, 1998.
3. M. Mateas. *An Oz-Centric Review of Interactive Drama and Believable Agents*. Technical Report CMU-CS-97-156, Department of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 1997.
4. J. Jull. A Clash Between Game and Narrative. *Proceedings of the Digital Arts and Culture Conference*, Bergen, Norway, 1998.
5. R.M. Young. Creating Interactive Narrative Structures: The Potential for AI Approaches. *AAAI Spring Symposium in Artificial Intelligence and Computer Games*, AAAI Press, 2000.
6. R. Bindiganavale, W. Schuler, J. Allbeck, N. Badler, A. Joshi, and M. Palmer. Dynamically Altering Agent Behaviours Using Natural Language Instructions. *Autonomous Agents 2000*, Barcelona, Spain, pp. 293-300, 2000.
7. M. Cavazza and I. Palmer. Natural Language Control of Interactive 3D Animation and Computer Games. *Virtual Reality*, 4:85-102; 1999.
8. V. DeLeon and B. Berry. Bringing VR to the Desktop: Are You Game? *IEEE Multimedia*, 7:2, pp. 68-72.
9. R.M. Young. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment. *AAAI Spring Symposium in Artificial Intelligence and Interactive Entertainment*, AAAI Press, *to appear*, 2001.
10. D. Robey, I Palmer, N. Chilton, J.

Dabeedin, P. Ingham, and S. Bramble. From Crime Scene to Computer Screen: The Use of Virtual Reality in Crime Scene Investigation. *Proceedings of the 7th UK VR-SIG Conference*, 2000.

11. N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics, Animation and Control*. Oxford University Press, 1993.
12. N. Badler, B. Webber, W. Becket, C. Geib, M. Moore, C.Pelachaud, B. Reich, and M. Stone. Planning and parallel transition networks: Animation's new frontiers. In: S. Y. Shin and T. L. Kunii, (Eds.), *Proceedings of Pacific Graphics '95*, pp. 101-117. World Scientific Publishing, 1995.
13. S. Bandi and D. Thalmann. Space Discretization for Efficient Human Navigation. *Computer Graphics Forum*, 17(3), pp.195-206, 1998.
14. K. Perlin and A. Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. *Proceedings of SIGGRAPH'95*, New Orleans (USA), 1995.
15. D. Kurlander and D. T. Ling. Planning-Based Control of Interface Animation. *Proceedings of the CHI'95 Conference*, Denver, ACM Press, 1995.
16. B. Webber, N. Badler, B. Di Eugenio, C. Geib, L. Levison, and M. Moore. *Instructions, Intentions and Expectations.* Technical Report, IRCS-94-01, Institute for Research In Cognitive Science, University of Pennsylvania, 1994.
17. N.J. Nilsson. *Artificial Intelligence: A New Synthesis*. San Francisco, Morgan Kaufmann, 1998.
18. R. Tsuneto, D. Nau, and J. Hendler. Plan-Refinement Strategies and Search-Space Size. *Proceedings of the European Conference on Planning*, pp. 414-426, 1997.
19. R. E. Korf. Artificial intelligence search algorithms. In: M.J. Atallah (Ed.), *CRC Handbook of Algorithms and Theory of Computation*, CRC Press, Boca Raton, FL, 1998, pp. 36-1 to 36-20, 1996.
20. N.J. Nilsson. *Principles of Artificial Intelligence*. Palo Alto, CA. Tioga Publishing Company, 1980.
21. J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading (Massachusetts), Addison-Wesley, 1984.
22. K. Knight and E. Rich. *Artificial Intelligence 2nd Edition*. McGraw Hill, 1991.
23. R.E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42:2-3, pp. 189-211, 1990.
24. J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. *Proceedings of SIGGRAPH'99*, Los Angeles (USA), pp. 29-38, 1999.
25. C. Geib and B. Webber. A consequence of incorporating intentions in means-end planning. *Working Notes – AAAI Spring Symposium Series: Foundations of Automatic Planning: The Classical Approach and Beyond.* AAAI Press, 1993.
26. E. Hansen and A. S. Zilberstein. Heuristic Search in Cyclic AND/OR Graphs. *Proceedings of the 15th National Conference on Artificial Intelligence*, AAAI Press, 1998.
27. I.L. Davis. Warp Speed: Path Planning for Star Trek: Armada. *AAAI Spring Symposium in Artificial Intelligence and Interactive Entertainment*, AAAI Press, 2000.
28. G. C. Stockman. A Minimax Algorithm Better than Alpha-Beta? *Artificial Intelligence*, 12, pp. 179-196, 1979.