"Situated AI" in Video Games: Integrating NLP, Path Planning and 3D Animation

Marc Cavazza, Srikanth Bandi and Ian Palmer

Electronic Imaging and Media Communications, University of Bradford
Bradford, West Yorkshire, BD7 1DP, United Kingdom
E-mail: {M.Cavazza, S.Bandi, I.J.Palmer}@Bradford.ac.uk

Abstract

This paper investigates the integration of Artificial Intelligence techniques into video games, from the perspective of voice controlled adventure games. Speech-based interfaces would add new dimensions to the control of video games. However, the integration of speech raises several issues, such as the adequate level of interpretation that should be carried by the system. We describe the implementation of a command interpreter based on Natural Language Processing techniques and specifically discuss the generation of appropriate system actions from spoken commands. The relevant AI techniques are plan generation and path planning. We consider the generation of simplified plans within the situated AI paradigm as a good candidate for integration with NLP. Work is in progress to reincorporate traditional search-based path planning into the same paradigm.

1. Introduction: Speech Control of Video Games

In this paper, we present ongoing research in the integration of AI technologies into real-time animation systems, which targets game applications. This work originated with the project of controlling through speech certain video games, such as the popular DOOM™ game. While speech input is the most user-friendly interface, it can also add new dimensions to the control of game characters. The use of speech control enables the user to concentrate on the visual scene rather than on the I/O devices and also brings increased realism. However, shifting from direct control to language-based control requires that the system possesses some interpretation capabilities that would correspond to the higher level of abstraction required. These interpretation capabilities should also compensate for the difference in response time between direct commands and spoken commands.

There has been previous research in the control of game-related actions through natural language input, such as that of Chapman (Chapman 1991). More recently, the AnimNL project at University of Pennsylvania (Webber et al. 1994) has explored some important issues in the relations between natural language instructions, actor behaviour and the actor's environment, though the intended system is not meant to be fully interactive. One important result is that natural language instructions can generate complex behaviours that are best described by planning techniques (Geib, Levison, and Moore 1994). However, in many systems implementing high-level control (mainly through natural language) of animated characters' behaviours described in terms of planning techniques tend to be implemented by simpler techniques. Plans are essentially compiled into scripts or Finite-State systems, good examples being the SODAJACK system (Geib, Levison, and Moore 1994) and the PERSONA project (Kurlander and Ling 1995).

From our early specifications, it became soon apparent that it was not possible to dissociate the Natural Language Processing (NLP) aspects from their interpretation into game playing functions. The latter are supported by two main techniques, which are planning and path planning. It is only after we implemented our NLP module that the full complexity of the integration of NLP with other AI techniques appeared. In the next sections, we will give an overview of the NLP analyser implemented. We will then discuss how its output can be interpreted to match the specific functions of a video game, and, after describing the implementation of a path planning system, draw conclusions for the integration of a fully functional voice-operated video game.

2. Natural Language Processing of gaming commands

To avoid biases in the definition of possible spoken commands, we used as a working corpus a compilation of DOOM™ spoilers available from the internet. These texts describe on a step-by-step basis the traversal of a game level, including the optimal paths and the various actions to be taken. At each stage, the recommended actions are described as specific instructions, such as "enter the door with the skull on it", "backup and wait for the door to close", "go back to the stairs", "shoot the Cyberdemon with a chaingun". At the semantic level, the distinction to be made is between commands that can be executed with minimal processing and commands that require extensive interpretation. Examples of the latter are *doctrine statements* and *purpose statements* (following Webber's terminology in (Webber et al. 1994)). Doctrine statements express general rules with no direct connection with the situation at hand, such as "shoot first, ask questions later" or "just battle away with rockets, chaingun, etc." As such they describe abstract generic knowledge, largely in the form of meta-rules that should guide system behaviour. Doctrine statements are certainly out of the scope of the current work. Purpose clauses describe actions that cannot be determined from the given instruction alone. One example in our corpus is "shoot a barrel to get rid of most of the pink demons". Proper processing of that command can only be achieved through plan generation, generating complex preconditions for action, such as the determination of demons and barrels' relative positions, etc. These are typically aimed at human subjects as they carry the justification or explanation for the action together with the action description. It should be noted that, apart from the technical difficulties, there is a practical argument against extensive interpretation by the system. If the system had indeed the capability to process high-level goals these would alter dramatically the nature of the game experience itself, and in many cases make it much less enjoyable. We will illustrate the necessary compromise between intelligence and playability in the next sections, after having described the NLP techniques implemented in our experiments.

2.1 An Integrated Parser for Spoken Commands

The first step of speech understanding is speech recognition, for which we use a commercially available system (Nuance6™ from Nuance Communications). We defined a recognition grammar for the system to ensure acceptable accuracy, thus placing ourselves in the paradigm of habitable command languages (Zoltan-Ford 1991). This is based on specifying a significant number of stylistic variant for a given concept such as to best cover the variability of utterances. In the specific case of video games, we expect that considering the benefits of speech control and the learning aspects that most games exhibits, the user would easily accept this constraint. The recognised sentence is then passed to the actual NLP module.

The NLP module is based on a parser integrating syntax and semantics. As evidenced from the study of DOOM spoilers, the various parameters for commands and actions call for syntactic processing capabilities. The need for syntax is for instance evidenced by the frequency of prepositional attachments used either in action parameters ("shoot the baron with the plasma gun") or in definite descriptions identifying world objects ("open the door with the skull on it"). The integrated parser is based on Tree Furcating Grammars (Cavazza 1998) as a variant of Tree adjoining Grammars (Joshi, Levy, and Takahashi 1975). This formalism is based on a an extended domain of locality and as such achieves a fair degree of syntax-semantics integration, which makes possible to construct a semantic representation for the command in a single step.

Figure 1 is an overview of the processing for the command "shoot the baron with the BFG9000". The various steps of processing can be described as follows:

- the elementary trees corresponding to each word are gathered into a forest (whenever one word can activate more than one tree, several forests are constructed)
- each forest is parsed left-to-right by checking possible tree combinations based on tree categories (initial or auxiliary trees)
- parsing is based on the combination of two adjacent trees, following substitution (S) or furcation (F) operations. This process is iterated until the forest is reduced to a single tree of sentential root or no further operations are possible
- synchronous semantic operation are carried together with tree fusion. For instance,

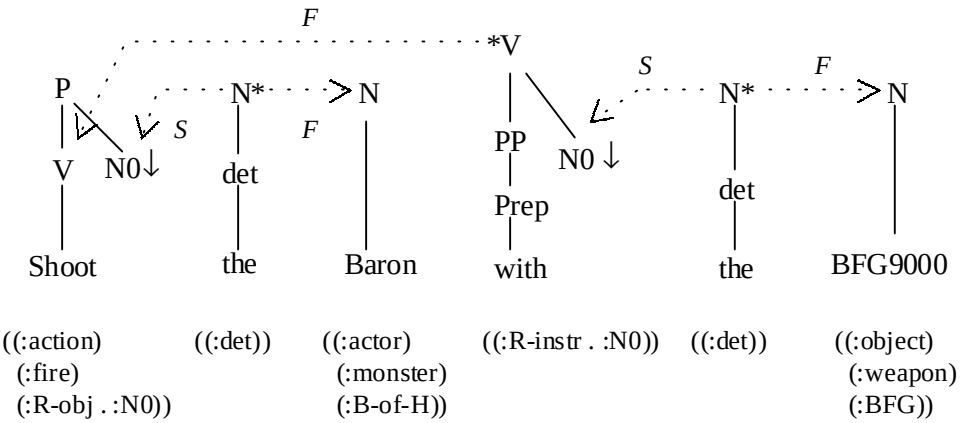

Fig. 1. Parsing a spoken command with the TFG formalism

substituting "baron" to N0 in tree "shoot" establishes a corresponding semantic relation (the baron is the *object* of the shooting action)

The output of the integrated parser is a feature structure with instantiated semantic relations corresponding to the various thematic roles. The next step is to generate corresponding actions from the characters in the form of a script of behaviours. Rather than using the DOOM engine, we have emulated its functions (fig. 2) within our REALISM animation system (Palmer and Grimsdale 1994). Another variant is that we choose to represent the player by an avatar, thus implementing third-person mode. This point has implications both in terms of command interpretation and path planning. The semantic structure produced from the analysis of the example sentence is:

```
(:action (:fire)
        (:R-obj ((:actor) (:monster) (:B-of-H)))
        (:R-instr ((:object) (:weapon) (:BFG))))
```

which has to be converted into a corresponding script, i.e. elementary system actions that are dictated by the nature of the game itself. The first conversion takes place within the NLP module itself:

```
((:select-weapon :BFG)
 (:target :baron)
 (:fire))
```

Instructions one and three correspond to DOOM commands that can be emulated, while instruction two involves checking lines of sight and adjusting direction. We will be assuming a kind of automatic target lock-on, as the field of vision is rather narrow.

Finally, this would correspond to the following code within the REALISM system:

```
selectItem("BFG9000");
if (target = visible("baron")) {
  rotateY(target->getPosition());
  useItem();
}
```

Other control structures are desirable, such as those repeating a given action until the goal is met. For instance, let us consider the difference in processing "shoot the imp with rocket launcher" and "kill the baron with the plasma gun". In DOOM, the intention is always to kill. While a single strafe of plasma gun will kill an imp, several shots (or a prolonged strafe) are needed to neutralise a baron.

The system has been implemented in Common LISP and runs on a O2 SGI workstation with a 150 MHz R10000 processor, the speech recognition system running on a NT PC. The main requirement for speech understanding in the context of video games is user real-time response. That is to say that the appropriate action should be taken by the system within 1000 ms of the end of the command. We do not address here "emergency" commands that could for instance bypass the NLP component and for which speech input is not optimal anyway. As the speech recognition process would take up to 500 ms,

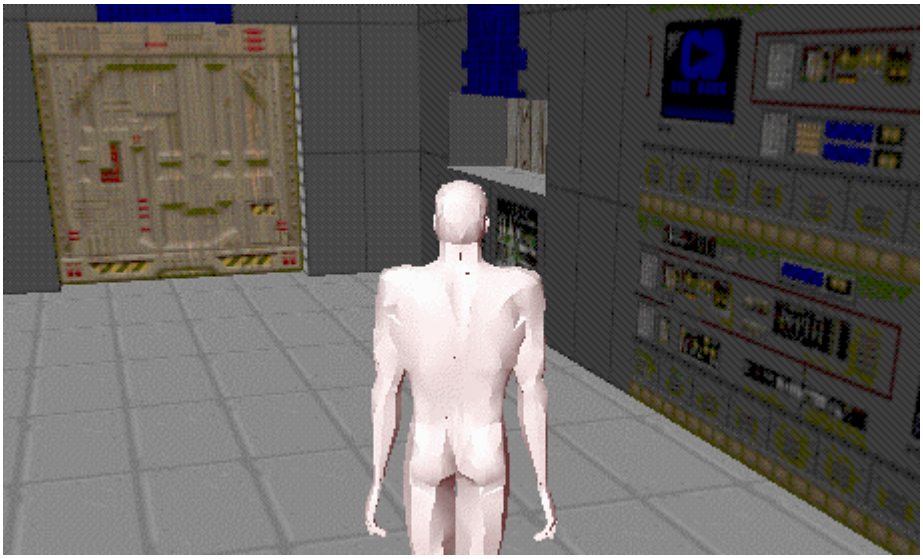

Fig. 2. Emulation of the DOOM environment with the REALISM system.

this gives an indication of what is left for the remainder of AI computations (Cavazza, Palmer, and Parnell 1998). The average time to process a command from the corresponding recognised ASCII string is 150 ms for even moderately complex orders such as those involving PP-attachments.

2.2 Command Interpretation

The variety of plans that can be generated in response to the interpretation of a sentence reflect the variability of situations encountered and the fact that commands are issued on the basis of the visible field of action. Considering even a simple command such as "shoot the imps with the chaingun", this might correspond to radically different situations depending on whether the player already possesses the chaingun or has it in his line of sight but should pick it up on his way to shoot the imps. Obviously, the parse tree and the explicit semantic representation are identical in both cases. It is the conversion of the semantic representation into a game script that differs both in content and in nature. The straightforward case consists in selecting the proper weapon and shooting, where shooting is automatically applied to the selected weapon. The latter situation calls for a quite different interpretation, as it involves generating a plan comprising the collection of the chaingun and the shooting action that should be delayed until the chaingun has been collected. We implicitly place the level of processing at the level of autonomy required for the system. We expect the system to make local decisions only, which have to be subordinated to a specific action such as path planning, shooting a target, picking-up an object. We thus consider that "pick up the stimpack" is a valid command, while "find a medikit" or "watch your health level" are not. In spite of the role potentially played by reference in voice-controlled video games, we have not considered multimodal approaches, where the player would specifically designate objects within the scene. This, for several reasons including the need for specific pointing devices and the fact that the line of sight roughly plays the role of a deictic zone.

Another consequence is that the main types of situations would have to be described a priori and explicitly stated in the spoken command. This would be a consequence of the situated nature of the possible actions within a given video game. The notion of situated action originally arises from the practical impossibility of defining or implementing successfully plans for certain specific situations. This leads to a change in the granularity of the actions considered and/or the control strategy applied (i.e. no control strategy apart from reaction to specific situations). In the field of games (but not specifically video games) this has been advocated by Agre and Chapman (Agre and Champan 1987) for the "Pengi" game. A strong context (such as the one of a video game which has a rich body of situation with specific tactic described) can facilitate the increase in granularity, which can be seen as characteristic of the situated approach (see e.g. (McCarthy 1998)). Our

central hypothesis can now be reformulated as follows: because of the limited initiative that the guided actor should have, it is possible to implement its behaviours through situated actions, by generating scripts from the NLP step. These descriptions would arise from an a priori study of the various games phases, which is largely facilitated by the existence of textual material about it. As a result, we would have specific scripts corresponding to such situated concepts as pick-up-visible-chaingun, shoot-visible-named-target, etc.

3. Path Planning for Artificial Characters

We find voice control of a video game to be more appropriate in a third person display mode. In this context, generating movements of the avatar in response to a specific situation is one of the most complex functions to be managed by the system. This section describes a grid-based approach for navigation of artificial human characters. In current game applications the environments are becoming increasingly realistic with the availability of high-speed graphics hardware. Hence any navigation algorithm should consider complex scene geometry for path finding. This should also run efficiently enough for real-time simulation demanded by game applications. The proposed methods in this paper address the problem of path finding in complex 3D interior scenes, which are especially appropriate for video games. In our current implementation, this assumes static scenes inhabited by a single actor. The first step consists in building an appropriate environment map. The 3D scene should thus be converted into a map searchable for optimal paths. A discretised 3D grid is adapted as the environment map because of the efficiency gains it offers. Sections of the scene are successively displayed on a raster screen and corresponding values read from the frame-buffer. From this, the occupancy status of the cells in the 3D grid can be determined. Since this is the most time consuming process, it is performed only once and hence the emphasis on static scenes.

The path finding is done between two cells in the grid. The basic method is “flood-fill” or “bushfire” algorithm used in filling closed regions with irregular and non-polygonal boundaries. The regions lie on 2D discrete cell grid. A seed cell is selected in the region and its neighboring cells are enumerated. In the next iteration the new cells in turn generate their neighbors without repeating cells enumerated earlier. The cells which are marked as obstacles are avoided. The enumerated cells expand like a “wavefront” in a breadth-first manner. All cells in a wavefront are marked with their iteration number. Once a goal cell is reached, the cells are traced back across decreasing values of wavefronts until the seed cell is reached. The path generated is optimal according to the Manhattan L1 metric. Several innovations are added to extend this algorithm for efficient 3D grid search (Bandi and Thalmann 1998).

Every cell in 3D has 26 neighbors as opposed to 8 neighbors in 2D. Thus a naive extension of basic 2D flood-fill to 3D grid will be grossly inefficient. However, a human actor walks only on the surface. This means only the cells lying on surfaces need to be enumerated. If (x,y,z) is a surface cell, then (x,y+1,z) is empty. With this definition, the effective cell neighbours reduce to 8 as in the 2D case. Hence performance of path finding in 3D scenes runs close to 2D levels. We use a simple A* algorithm for path expansion. Instead of expanding all the cells into a breadth-first tree, a heuristic is applied to select cells closest to the goal for next expansion. An example of a generated path in a 3D environment is represented on figure 3.

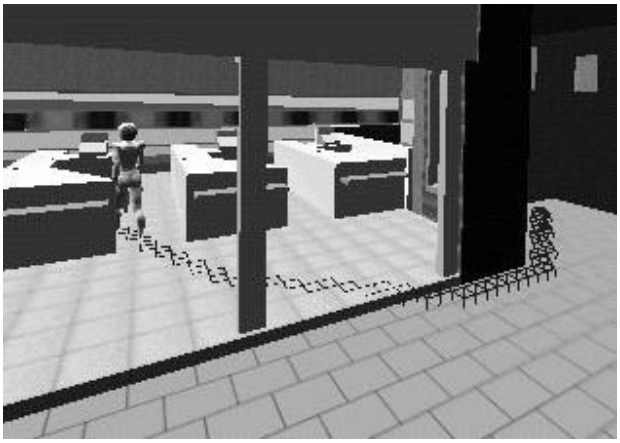

Figure 3 . Human-like path planning (graphic layout courtesy of LIG-EPFL)

By treating objects as obstacle or non-obstacle, common-sense meaning may be missing from the scene data. For example, following optimal path criteria, the actor may step on tables even though it is not socially correct to do so. This semantic information can be encoded in cells and used in path

planning. It would eventually relate to proper common sense knowledge representations or an ontology of physical behaviours.
In many games we see exotic actions such as panic stricken actors leaping from one location to other disconnected regions to ward of dangers. Traditional flood-fill requires the cell regions to be connected. By expanding the concept of a cell neighborhood, disconnected cell regions could be taken care of. This means, instead of enumerating only immediate cell neighbours, cells separated by empty spaces could be deemed to be neighbors, if they are sufficiently close. The "closeness" or "reachability" depends on human physical dimensions. A child cannot cross empty spaces in the same way as an adult can. If an enumerated neighbor requires a jump or a leap from the current cell, this information can be encoded in the current cell while generating the path. When the actor reaches this cell during a walk, appropriate action indicated by the cell is carried out, thus allowing some reactive behaviour. It is possible to have several alternate paths at a given cell. For example, human model may be presented with a bridge and a tunnel path underneath, the choice is resolved by prioritizing the actions required to take alternate paths. If the tunnel path requires a normal walk and the bridge requires a jump, the tunnel is selected. In this case jumping is assigned lower priority since it requires greater energy expenditure. In case of identical actions the tie resolution is unpredictable. Usually physical dimensions of human come into play in such situations. If a tunnel is to small, only a human of short height could pass through. Taller human models invariably select the bridge.

Though the environment is assumed to be static, small changes are permitted. The modifications required in the cell path are assumed to be small compared to the total length of the path. This permits local, efficient updating of the path. Finally, the cells paths are usually distorted due to the L1 metric used in distance computations. The crooked paths can be smoothened by computer graphics techniques such as a Digital Differential Analyzer (DDA) employed in rasterising straight lines on graphics screens. Another aspect is to identify border regions in a scene so that the actor does not walk unnaturally close to the obstacles.

The path finding takes place in real time except when dynamic scene changes are allowed. When scene changes, that portion must be re-discretised to update the cell grid.

Path planning is involved in the interpretation of many spoken commands such as "go back to the stairs" (see above). In some cases, the semantics of the spoken command influences the choice of the path itself. For instance, the specific instruction "run through the radioactive slime to the red door" might go against default values for terrain cost when using A*-based path planning. In which case part of the semantic interpretation would have to alter the cost values for some specific terrain as well. Finally, we have just been dealing with path planning in a static environment. At some stages, it might be interesting to explore the inclusion of path planning for tracking moving targets, using e.g. variants of LRTA* (Ishida and Korf 1995).

4. Conclusions

While the AI techniques described in this paper have all been implemented, we have so far only achieved partial integration of these technologies within the gaming environment. Namely, we have been investigating the constraints of real-time performance while integrating NLP with animation techniques in the context of the DOOM emulator (Cavazza, Palmer, and Parnell 1998). Our next step is to achieve a similar degree of integration with path planning as demonstrated in (Bandi and Thalmann 1998). Starting from the requirements of user interaction within a clear scenario provides useful guidelines for such an integration, however. As the natural development for path planning techniques is to include some of the environment semantics, the requirements of realistic situations also constitute a useful starting point. In gaming the guided actors should display a limited amount of intelligence otherwise the challenging aspects of the game would disappear. This provides an ad hoc solution to the problem of generating plans from natural language instructions, which in itself is representative of the inclusion of genuine AI techniques into video games. In our view, the necessary integration of AI techniques in video games is faced with a difficult compromise between the sophistication of the algorithms implemented, their integration with computer graphics techniques and the requirements for user real-time performance.

Acknowledgments

Work on path planning was carried by one of the authors (SB) while at Computer Graphics Lab (LIG), Ecole Polytechnique Fédérale de Lausanne.

References

Agre, P. and Chapman, D. 1987. Pengi: an Implementation of a Theory of Activity. *Proceedings of the Sevent National Conference on Artificial Intelligence*. Menlo Park, Calif. AAAI Press.

Bandhi, S. and Thalmann, D. 1998. Space Discretization for Efficient Human Navigation. *Computer Graphics Forum*, 17(3):C195-C206.

Cavazza, M. 1998. An Integated TFG Parser with Explicit Tree Typing, *Proceedings of the Fourth TAG+ Workshop*, Technical Report, IRCS-98-12, Institute for Research in Cognitive Science, University of Pennsylvania.

Cavazza. M., Palmer, I. and Parnell, S. 1998. Real-Time Requirements for the Implementation of Speech-Controlled Artificial Actors. In: N. Magnenat-Thalmann and D. Thalmann (Eds.), Modelling and Motion Capture for Virtual Environments, Lecture Notes in Artificial Intelligence 1537, New York, Springer.

Chapman, D. 1991. Vision, Instruction and Action. Cambridge, MIT Press.

Christopher Geib, Libby Levison, and Michael B. Moore. 1994. *Sodajack: an architecture for agents that search for and manipulate objects*. Technical Report, MS-CIS-94-16, Dept. of Computer and Information Science, University of Pennsylvania.

Ishida, T., Korf, R.E. 1995. Moving-Target Search: A Real-Time Search for Changing Goals, *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 6:609-619.

Joshi, A., Levy, L. & Takahashi, M., 1975. Tree Adjunct Grammars. *Journal of the Computer and System Sciences*, 10(1)136-163.

Kurlander, D. and Ling, D. T. 1995. Planning-Based Control of Interface Animation. *Proceedings of the CHI'95 Conference*, Denver, ACM Press.

McCarthy, J., 1998. Partial Formalisations of the Lemmings Game. **http://www-formal.stanford.edu/jmc/lemmings/lemmings.html**

Palmer, I.J, Grimsdale, R.L. 1994. REALISM: Reusable Elements for Animation using Local Integrated Simulation Models, Proceedings of Computer Animation'94, 132-140. IEEE Comp. Soc. Press.

Webber, B., Badler, N., Di Eugenio, B., Geib, C., Levison, L., and Moore, M. 1994. *Instructions, Intentions and Expectations*. Technical Report, IRCS-94-01, Institute for Research In Cognitive Science, University of Pennsylvania.

Zoltan-Ford, E. 1991. How to get people to say and type what computers can understand. *The International Journal of Man-Machine Studies*, 34:527-547.