

AI-based World Behaviour for Emergent Narratives

Jean-Luc Lugin
School of Computing
University of Teesside
TS1 3BA Middlesbrough
United Kingdom
j-l.lugin@tees.ac.uk

Marc Cavazza
School of Computing
University of Teesside
TS1 3BA Middlesbrough
United Kingdom
m.o.cavazza@tees.ac.uk

ABSTRACT

Research in Interactive Narrative has developed new approaches to the behaviour of virtual actors, but has dedicated little attention to the physical behaviour of the environment in which the action takes place. In this paper, we describe a method supporting the AI-based simulation of object behaviour, so that interactive narrative can feature the physical environment inhabited by the player character as an “actor”. The prototype we describe has been developed on top of the Unreal Tournament game engine and relies on a “causal engine”, which essentially bypasses the native Physics engine to generate alternative consequences to player interventions. It operates using a small depth-bound planning system which determines the most appropriate object behaviours following player interaction. The prototype is illustrated through a test application called “Death Kitchen”, freely inspired from various thriller and horror films, in which the kitchen is plotting against the player character to generate domestic accidents.

Categories and Subject Descriptors

H5.1 [Multimedia Information Systems] Artificial, Augmented and Virtual Reality -Virtual Reality for Art and Entertainment.

I2.8 [Artificial Intelligence] Problem Solving, Control Methods, and Search

General Terms

Design and Experimentation.

Keywords

Interactive Narrative, Artificial Intelligence, Causal Perception.

1. INTRODUCTION AND RATIONALE

Most computer games apply a clear separation between the various behavioural elements that support gameplay. Physics is traditionally associated to the evolution of world objects (generally in reaction to user interaction), while AI is used to support the behaviour of agents and action selection. The balance between the various behavioural aspects varies according to the game genre, often in a characteristic fashion. Even new paradigms

of computer entertainment such as Interactive Storytelling have, in their own way, maintained a strict distinction between physical aspects of the game and the behaviour of virtual actors within the narrative. Despite the fact that various levels of causality have been described in storytelling (physical, psychological and narrative) these have not been fully exploited in the search for new entertainment experiences [4]. Over the previous years, we have revisited this distinction by exploring the use of AI techniques to support the behaviour of a whole physical environment [5] [6].

This approach has a specific interest for entertainment applications, in particular the possibility of re-incorporating physical behaviour under the direct control of storyboarding. Instead of scripting all possible interactions, these could be derived dynamically from first principles, linked to scenario design. In a similar fashion to Interactive Storytelling, this brings the possibility of dynamic, emergent story generation from the physical behaviour of the environment.

Previous research has addressed, from various perspectives, the definition of high-level behaviours for objects in relation to user interaction. Kalmann and Thalmann [9] have introduced the notion of “smart objects” whose behaviours would adapt to different modes of interaction, essentially from the perspective of object functionality. More recently, Abaci et al. [1] have suggested that object behaviour could support problem solving using planning techniques. Garcia et al. [8] have addressed the issue of discrete simulation which is an important notion in the transition from physics-based behaviour to AI-based behaviour (although not strictly speaking the focus of their work). In previous work [11] we have introduced a “causal engine” that creates event co-occurrences from interactions with objects. This system relies on a discretisation of events, which are analysed in terms of high-level actions, and the properties of objects they involve, in order to determine alternative consequences creating illusions of causal perception. The ScriptEase™ system [13] which has been developed as a generic framework for behaviour authoring includes encounter patterns, which generate scripts associated to inanimate objects: this supports in particular high-level behaviour in reaction to a situation. Finally, Willans and Harrison [16] have introduced formal models of interactive object behaviour using Petri Nets. Interestingly, their work has described interaction with kitchen appliances in a virtual world, which constitute a kind of environment we previously investigated as well [5] and, under a revised form, will serve as the setting for the experiments reported in this paper. Overall, there is a growing interest in describing high-level object behaviours to support more complex problem solving or the “procedural” generation of interactive content. One challenge appears to be the integration of

such smart object behaviour into an overall environment behaviour supporting narrative or gameplay.

In the next sections, we describe the technology baseline for our prototype which controls objects behaviour from a global perspective, using a real-time analysis of potential situations. We introduce the mechanisms for event interception and situation recognition, as well as the generation of system actions using limited-depth search-based planning (inspired from [3]) The system behaviour will be illustrated through practical examples.

2. SYSTEM OVERVIEW

Our experimental prototype, “Death Kitchen” presents itself as a small 3D interactive narrative, whose scenario has been freely inspired from films such as “Final Destination”. The overall plot consists in a reactive environment which maximises the level of danger for the user, clearly attempting at inflicting the maximum level of damage to the player’s avatar. At this stage, we have not defined a precise gameplay, which would for instance see the user trying to survive as long as possible. We have left options open so that the system could be eventually demonstrated as a narrative/simulation game (featuring autonomous agents) or as a third-person game (notwithstanding other simulations and “serious games” applications).

The environment models a large-scale kitchen, large enough for the user to navigate through different areas. The kitchen is complete with traditional appliances such as gas hobs, microwave oven, coffee machine, cutlery, etc. It is furnished with cupboards

and drawers containing additional objects such as bottles, etc.

The environment essentially reacts to the player character interactions. These can be of two kinds: the first one consists of interactions with world objects or appliances, which constitute specific affordances: the avatar can carry out a few actions on an appliance as an object (physical manipulation such as grasping) or as a device (using the appliance or activating it). The second one is based on the avatar’s navigation in the kitchen environment, which brings it in different zones in proximity of, or away from, hazardous situations. These reactions consist in triggering events and object behaviours dynamically computed from an assessment of the current situation to maximise the threat to the user. In most cases, these events could appear as a consequence of user interaction, in virtue of users propensity to attribute causal relations between co-occurring events, and this plays an important role in the impression of emergent narrative.

The generation of object behaviours by the environment is based on an AI technique recognising situations created by the user and analysing their potential for creating threats. This AI module bypasses the native Physics engine to control the unfolding of actions. Default consequences of actions, which would normally be passed to the Physics engine, are “frozen” during a sampling cycle until the system has determined the most appropriate consequence. Our system architecture is composed of an AI module called Causal Engine acting upon the Unreal Tournament game Engine [10]. As shown of figure 1, the Causal Engine, developed in C++, constantly intercepts and modifies actions

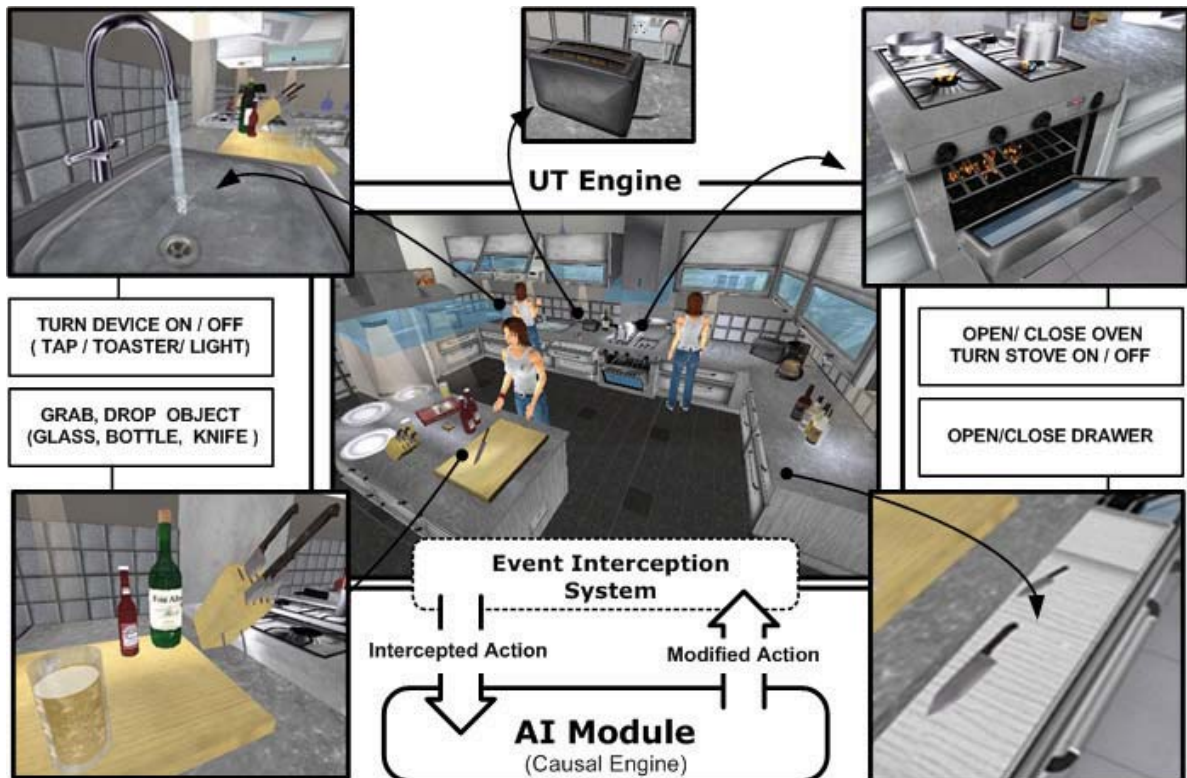


Figure 1. Environment and System Overview showing potential navigation of the player character through different areas containing various types of appliances and objects.

occurring in an Unreal Environment through a special interface named Event Interception System (EIS for short). This module directly built-above the visualisation native event system is responsible for the action recognition, interception and reactivation [12]. The overall cycle is composed of five phases i) Interaction Event Interception ii) Action recognition iii) Situation assessment iv) Selection of Environment Reaction and v) Effect Generation.

During the first phase, the system samples graphic events corresponding to user interactions with the world's objects or to interaction between objects triggered by the user (such as collisions between objects). Its sampling timeframe is on average 10ms. Some of these Basic Events, such as USE or GRASP, refer to events triggered by the user when controlling the avatar to intentionally interact with the environment. Other Basic events like HIT, TOUCH, IN, OUT, ON are derived from UT event system primitives such as bump, kImpact. They are defined as a combination of system primitives and tests on object properties and physical parameters of the impact such as momentum, etc.

In the second phase, the system tries to instantiate an action representation from the Basic Events recognised during the previous step. As described in the forthcoming section, Actions are defined as representations named CE (Cause-and-Effect

representation).

In the third phase, the situation created by the last user intervention is assessed in term of the risk they can imply for the user (i.e. the user's character). Here, the system produces a first pre-selection, on the set of actions recognised. From the nature of the actions' consequences, the system determines if they can directly result in the avatar getting hurt, in which case their effects, which had been suspended during action recognition, are directly reactivated in the environment without further processing. On the other hand, actions that have no immediate potential for harming the user are transformed to increase their level of danger before being re-activated. Indeed, the AI module replaces their natural consequences, when semantically possible, with those of actions increasing the level of risk of a certain spatial area around the user. As described in the following section, the AI module generates alternative effect to an action and assesses/plans the degree of risk of each of them according to the immediate user surroundings.

3. AI TECHNOLOGY BASELINE

The environment behaviour, consisting of system reactions to avatar actions, is under the control of an AI system that determines in real time the most appropriate system response, which is then triggered in the form of an object behaviour. It does

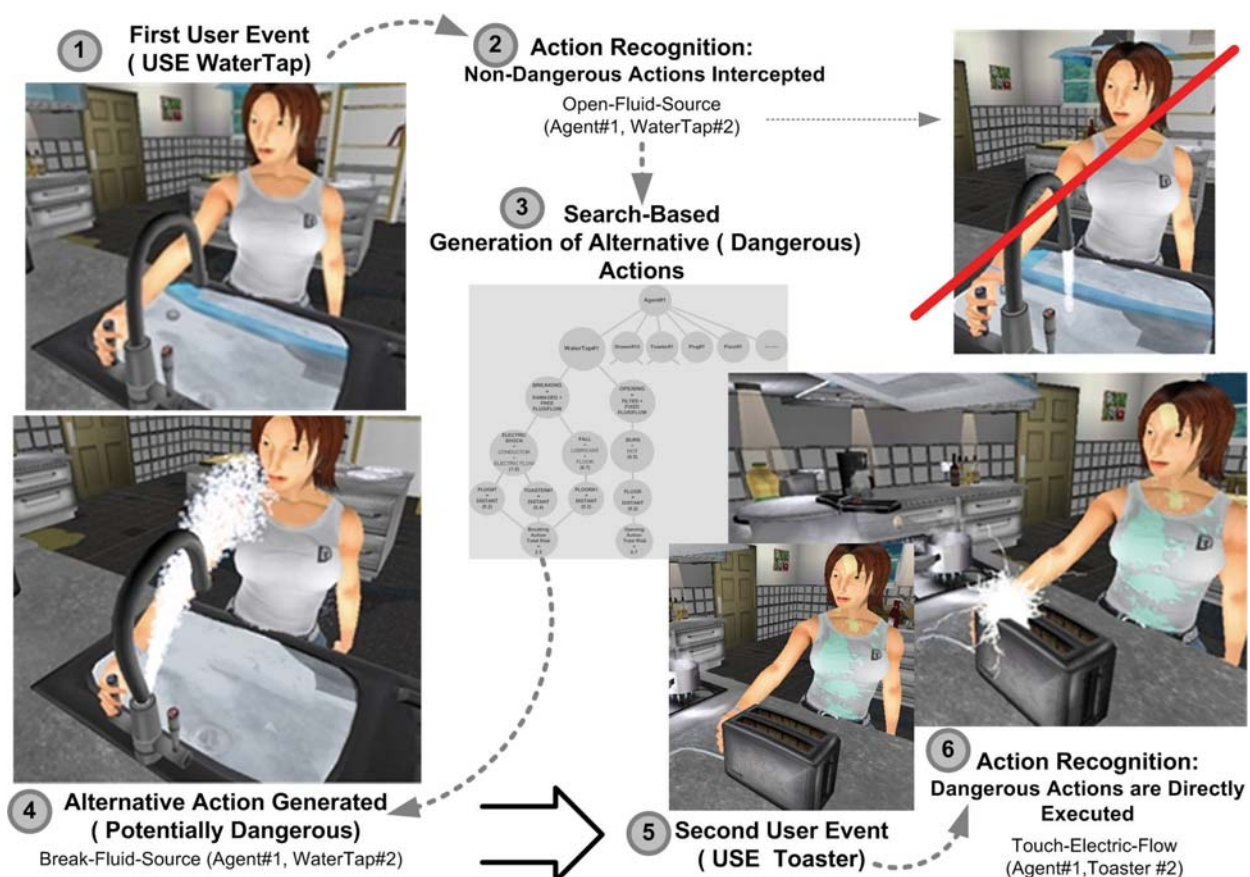


Figure 2. An Example of Hazardous Action Generation.

so through a search-based planning step, which searches the space of possible operators to be applied to the current situation at a limited depth (2 or 3).

This system's basic approach has been derived from our previous environment for creating "causal illusions" [6]. This environment is based on a real-time recognition of actions which are intercepted and have their normal consequences suspended to allow for the creation of alternative consequences by the system. For instance, in such a system, the impact of a falling glass on a table which would normally result in the glass shattering may trigger alternative consequences, such as the glass landing softly, while another glass standing on the table shatters without having been hit by the former. Action recognition operates by intercepting low-level physical events (such as object collisions) and interpreting them in terms of potential high-level actions (such as breaking). The whole system relies on a representation for actions that associates pre-conditions to action consequences [17], constituting an action ontology. Action transformation is driven by a search process, which applies action transformation operators using heuristics based on the level of disruption between the alternative consequence and the default one.

In the current system, we have retained the basic principles and software components of our "causal engine", with some significant differences, which have led to new developments. The first is that this new system is centred on the player character actions, rather than being only driven by interaction between objects. This has an impact on the granularity of some action representations. The second is that alternative consequences of the player actions are not based on measures of similarity with default

consequences but on gameplay-dependent heuristics. In our scenario, these heuristics aim at maximise the hazardous nature of the environment.

Although causality could have been represented using different formalisms, such as Bayesian networks [14], search-based planning offers a better support to real-time intervention on event causality. In particular, it can explore the space of alternative actions when coupled with appropriate object and action knowledge and has the long-term potential of generating more "creative" solutions, for instance if coupled with analogical reasoning.

The system relies on an ontology of possible actions and, more traditionally, an ontology of objects. Objects descriptions are centred on potential object states, whether these states be physical states (position, integrity) or functional states (a gas hob being turned on or off). Reasoning on object properties and action consequences would normally require a great deal of common sense knowledge, whose complexity many not be compatible with real-time performance. In our current implementation, this has been addressed through a specific formalisation of object properties using functional reasoning [2] [6] [15]. Functional representations relate physical properties to possible object behaviours: for instance for a container wall integrity and position determine filling/emptying behaviours. To a large extent this amounts to defining a fixed level of granularity for common sense reasoning, thereby facilitating knowledge elicitation as well as inference.

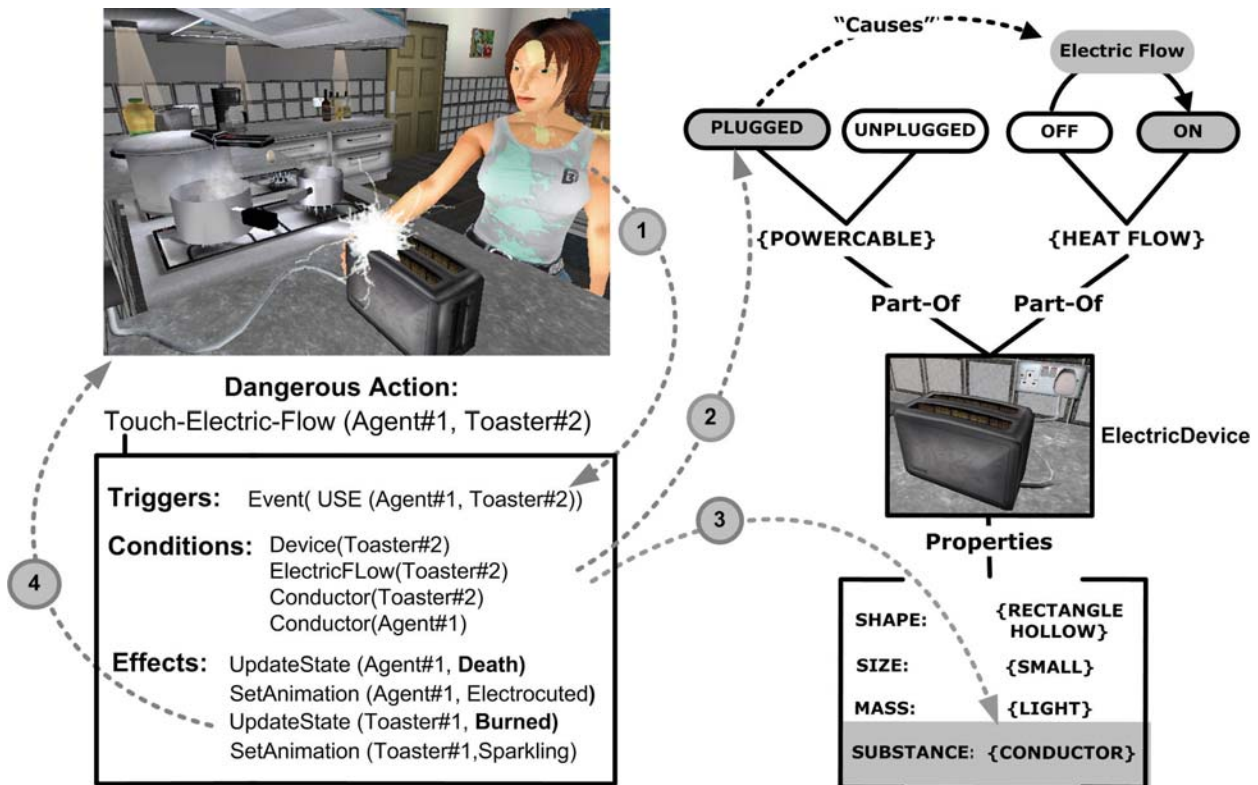


Figure 3. The Role of Action Representations in System Behaviour.

4. HAZARDOUS ACTIONS GENERATION

Figure 2 describes an example of system behaviour creating a hazardous situation in response to the player character's actions. In the first instance the player is moving to the sink to get tap water (this can be prompted by some element of the game play such as the need for food and water). The system intercepts the default interaction event (Figure 2-2) to generate an additional consequence (Figure 2-3) by which the tap sprinkles water on the character (Figure 2-4). This is at that stage the best action proposed by the system: it does not create any danger in itself but has potential for future hazardous situations. Further in the session the player may interact with a toaster (Figure 2-5; this again can be prompted by gaming instructions such as the need to obtain food, triggering an exploration of the environment). The character is soaked (from its previous "encounter" with the tap) and tries to turn on an electrical device made of a conductor substance (Figure 3), consequently the player character should receive an electric shock (Figure 2-7).

This system reaction is the result of the search process described in figure 5 (progressive construction of hazardous situations)

which makes use of the common sense representations for actions and objects. The planning system which aims at generating hazardous consequences operates by analysing the situation, using common sense knowledge on actions and objects. The object representation for the toaster groups together functional knowledge (on the object as an electrical appliance, Figure 3-2) as well as basic physical properties. This representation is used as part of the search process to determine the prospects of using electricity against the character. This is achieved using knowledge about the current situation (i.e. the fact that the character has been soaked by a previous action of the system) and relating the object properties (toaster, Figure 3-3) to relevant set of hazardous situations, defined in a matrix as explained in the following section. The search process (Figure 5) determines a possible system attempt at creating an electric shock, which is reflected in the final action representation prior to reactivation of the action consequences (Figure 3-4).

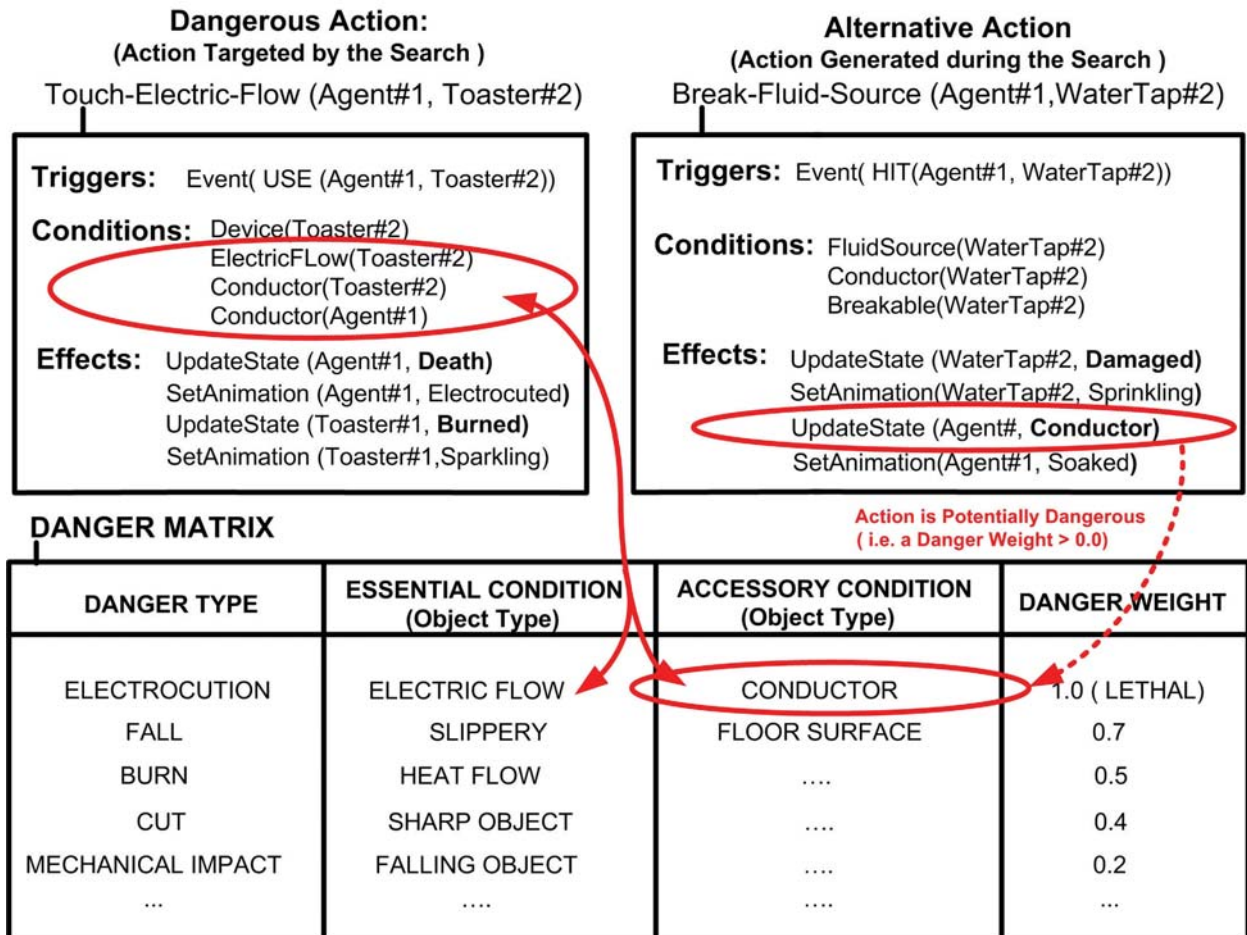


Figure 4. The "Danger Matrix" (D-Matrix) describes potential situations leading to domestic accidents. It is used during the search process to select actions based on previous events. After the first alternative action has been generated (sprinkling water on the character), the D-Matrix determines which consequences to the player action (using the toaster) should be triggered (generating an electric current).

5. ACTION GENERATION ALGORITHM

The main purpose of the AI module controlling the environment's behaviour is to transform any player action into a potentially dangerous one. It operates by intercepting in real-time the consequences of the player interaction in the environment. From an intercepted action and the current state of the world (defined by the state of objects in the avatar's proximity), its main algorithm dynamically generates and evaluates a set of "danger-generating" actions. The algorithm is based on limited-depth search-based planning and uses a "Danger Matrix" to guide its search in the space of consequences generation. The principles behind our "Danger Matrix" are that most domestic accident emerges from the conjunction of a set of hazardous factors within a close environment. This matrix (henceforth "D-Matrix") identifies potentially dangerous situations by associating a level of danger to situations, defined as object configurations. As shown in Figure 4, situations are described as a conjunction of primary and enabling conditions. For instance, a direct contact with a source of electricity¹ is the essential condition for an electrocution. However, an electric shock can also be propagated

through an electricity conducting substance (such as metal, or conducting fluid). In our representation we named the presence of a conducting substance the Accessory condition for that hazardous situation. This notion of accessory condition (or enabling condition) is essential to our approach, as it reflects the progressive building of dangerous situations, which is a central element of the gameplay/narrative.

The D-Matrix associates a "level of danger" to these situations, which can be used to guide the heuristic search of action transformations. The level of danger is a normalised coefficient, where 1.0 corresponds to a life-threatening event (such as electrocution) 0.0 represents an action posing no immediate danger (such as opening a fluid source). Intermediate values are defined depending on the severity of injuries potentially inflicted to the player's avatar (which mirrors the traditional representation of "health levels") The D-matrix is used throughout the process to evaluate the danger of the alternative effects considered by the search algorithm and that of the resulting situation.

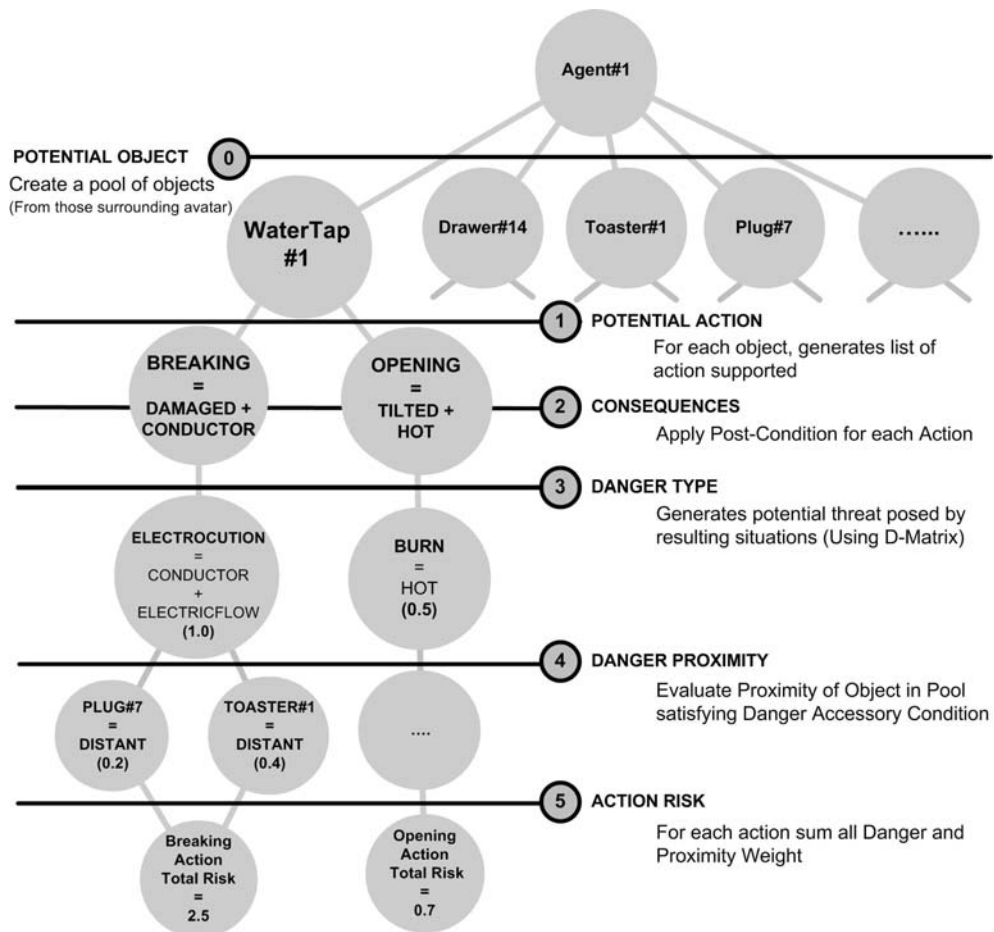


Figure 5 Search-based Hazardous Action Generation.

¹ Referred to as an "electric flow", due to the specific implementation of physical processes in the system

The generation and evaluation of the alternative consequences of the player actions is performed using limited-depth search-based planning. The search proceeds forward from the current situation and constructs sequences of operators whose purpose is to generate situations that would fall under the categories described by the D-Matrix. The search process remains tractable for a real-time system, because of the limited depth considered and the additional use of spatial proximity heuristics to guide the search process. We will illustrate the search procedure using the example previously introduced in figure 2.

As shown of figure 5, the search process comprises six steps: i) Object pre-selection ii) alternative action generation iii) Potential danger evaluation iv) Cost/Reachability of the situation v) overall risk evaluation, and finally vi) Action Selection. The search process is part of a cycle reinitialised every time a player action is recognised by the EIS module and not considered as directly dangerous (for instance, the user trying to open the tap to get water).

The first step in the process consists of the creation of a pool of candidate objects (Figure 5-0) for those present within a given radius of the player's character (these objects can correspond to kitchen appliances with functional properties or simpler objects such as knives, bottles, etc.). The objects selected are sorted according to their distance to the avatar. In order to preserve the system's real-time response, the selection is initially limited to ten objects. Restricting the objects to the player character's immediate surroundings also helps to suggest a causal relation between the avatar initial action and the observed reaction of the environment (in virtue of principles of causal attribution [6]).

Object selection thus constitutes the first level of the search tree. The search procedure progresses in forward fashion from each object selected to establish a set of alternative effects involving these objects, and their associated risks if executed in the environment.

The second step (Figure 5-1) generates all possible actions each object can be involved in. This step filters actions starting in the first instance with those corresponding to an object's functionality (e.g. a toaster or a gas hob may be turned on). As any physical object can take part in some elementary actions (such as falling, tilting over or sliding), there is a need to limit the number and type of actions considered. This is achieved in part by considering the contents of the D-Matrix.

Next, the search algorithm considers the consequences (or post-conditions) of the actions previously generated for each object. For instance, liquid containers breaking or tilting over will spill out their contents, which could create electricity-conducting areas (water), flammable areas (alcohol), slippery areas (cooking oil) all of which can form part of dangerous situations in the presence of appropriate conditions (Figure 5-2). The algorithm tries to map post-conditions onto essential or accessory conditions for a hazardous situation, as defined in the D-Matrix. A successful mapping will correspond to a potentially dangerous situation being generated by that search path, (Figure 5-3).

The next step, (Figure 5-4) re-assesses the environment to determine whether a given situation can practically occur considering the spatial distribution of objects for the primary and secondary danger conditions. It evaluates objects' position /distance relative to the user, and/or to the object completing the secondary danger condition as per the D-Matrix (for instance, a

conducting object complementing an electricity source). The proximity between objects is interpreted using a discrete categorisation merging both distance and orientation: TOUCH, BELOW, ABOVE, NEAR, and DISTANT (when no "realistic" configuration can be envisioned for this object). Finally, when proximity weights have been attributed to all the hazardous situations potentially created by an action, the system computes the total risk of an action by combining its danger and proximity weights (Figure 5-5).

The first step of the search (object generation) is not guided by any heuristics and in the first instance the algorithm is biased towards "depth-first" processing, in the sense that it completes the evaluation of situations potentially emerging by the object currently under consideration. To preserve the system's response time, in case no suitable objects have been identified within a given time sample, the algorithm can exit the search after having processed an object, if the search has exceeded a threshold of 100 ms to preserve causal attribution [6].

On the other hand, the search process can be successfully terminated if one the alternative actions has generated a risk superior to 1.0 meaning that at least one action will lead to life-threatening injuries). This derives from the need to achieve a good balance between response time and solution's interest, rather than producing an exhaustive set of all possible consequences.

For the moment we estimate the narrative quality of a plan in term of its plausibility regarding the degree of causal attribution associated to the alternative consequences triggered in response to user interaction. In our current prototype, the level of plausibility is related to the concept of level of Disruption. This normalised value (in the [0,1] range) influences the computation of the *proximity weight* during the search process. For instance, with a high level of disruption, close to 1.0, the search will consider DISTANT objects for possible influences, not restricting such influence to NEAR objects.

On the other hand, the quality of the plan generated depends to a large extent on the spatial, functional and physical property of the object surrounding the user. In our current environment, the user can interact with a vast number of objects in different ways (users can activate, move, project over 300 objects of 30 types). Such variations force the system to initiate a new search process for each user interaction. Therefore to increase the likelihood of creating on hazardous situation, the search constantly evaluates the user's current context. However, when the system is not capable of providing a potential hazardous action within a given context, the default action is reactivated. Indeed, time constraints and the richness of the environment in term of possible alternative actions can force the system to exit with no plausible plan. Then, every time the system fails to improve the risk of the environment, the *level of disruption* is increased. The variation of the level of disruption has two effects; first it provides a simple mechanism to generate a diversity of plans from the same initial situation. Secondly, its permits to envisage less realistic plans, and thus create more surprising situation for the user.

6. CONCLUSIONS

We have described an emergent storytelling prototype, in which the environment itself plays the role of a feature character. As a physical environment, its behaviour would normally be conceived of in terms of Physics. However, as a “feature character”, its apparently intentional actions are best accounted for by AI techniques. In a sense, we have extended/adapted current techniques of Interactive Storytelling, in which virtual actors actions are determined in real-time through AI Planning systems, to the animation of a set of physical objects. Another similarity with Interactive Storytelling is that the system maintains a constant awareness of the situation at hand, which is described in high-level terms. At this stage, the system is not yet equipped with intelligent camera control that could enhance the narrative experience.

A natural evolution of this work would be to re-incorporate it within an Interactive Storytelling system that would provide high-level control over the plot and the role of virtual actors. In a sense this would reconcile the various determinants of causality in Interactive Storytelling [4].

7. ACKNOWLEDGMENTS

Sean Crooks is thanked for developed part of the environment used on figure 1-2-3-4.

8. REFERENCES

- [1] Abaci, T., Cíger, J. and Thalmann, D. Planning with Smart Objects. *International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG) 2005*, 25-28.
- [2] Bicici, E. and St. Amant, R. Reasoning about the functionality of tools and physical artifacts. *Technical Report TR-2003-22*, Department of Computer Science, North Carolina State, 2003.
- [3] Bonet, B. and Geffner, H. Planning as heuristic search. *Artificial Intelligence: Special Issue on Heuristic Search*, 129, 2001, 5-33.
- [4] Cavazza, M., Charles, F., and Mead, S.J. Narrative Representations and Causality in Character-Based Interactive Storytelling. *Proceedings of CAST 2001*, Bonn, Germany, 2001, 139-142.
- [5] Cavazza, M., Hartley, S., Lugin, J.-L. And Le Bras, M. Qualitative Physics in Virtual Environments. *ACM Conference on Intelligent User Interfaces (IUI 2004)*, ACM Press, 2004, 54-61.
- [6] Cavazza, M., Lugin, J.-L., Crooks, S., Nandi, A., Palmer, M. and Le Renard, M. Causality and Virtual Reality Art. *Proceedings of the 5th conference on Creativity & Cognition*, ACM Press, 2005, 4-12.
- [7] Chandrasekaran, B. and Josephson, J.R. Representing Function as Effect, *AAAI-96 Workshop on Modeling and Reasoning about Function*, Portland, OR, August 1996.
- [8] Garcia, I., Molla, M., and Morillo, P. From continuous to discrete games, *Computer Graphics International 2004 (CGI'04)*, 2004, 626-629.
- [9] Kallmann, M. and Thalmann, D. Modeling Behaviors of Interactive Objects for Virtual Reality Applications, *Journal of Visual Languages and Computing*, 13(2), 2002, 177-195.
- [10] Lewis, M and Jacobson, J., Games Engines in Scientific Research, *Communications of ACM*, Vol. 45, No. 1, January, 2002, 27-31.
- [11] Lugin, J-L., Cavazza, M., Palmer, M. and Crooks, S. AI-Mediated Interaction in Virtual Reality Art. In: M. Maybury, O. Stock, W. Wahlster (Eds.) *Intelligent Technologies for Interactive Entertainment: First International Conference (INTETAIN 2005)*, Madonna di Campiglio, Italy, LNAI n. 3814, Springer-Verlag, 2005, 74-83.
- [12] Lugin, J-L. ,Libardi, P. Barnes, M., Le Bras, M. and Cavazza, M. Event-based Causality In Virtual Environment. *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, the Netherlands, 2004.
- [13] McNaughton, M., Schaeffer, J., Szafron, D., Parker, D. and Redford, J. Code Generation for AI Scripting in Computer Role-Playing Games. *AAAI-04 Workshop on Challenges in Game AI*, San Jose, USA, July 2004, 129-133.
- [14] Pearl, J. *Causality: Models, Reasoning and Inference*, Cambridge: Cambridge University Press, 2000.
- [15] Vaina, L. and Jaulent, M. Object structure and action requirements: A compatibility model for functional recognition. *International Journal of Intelligent Systems*, 6, 1991, 313-336.
- [16] Willans, J.C. and Harrison, M.D. A toolset supported approach for designing and testing virtual environment interaction techniques. *International Journal of Human Computer Studies*. 55(2), 2001, 145-166.
- [17] Wilkins, D. E. Causal reasoning in planning. *Computational Intelligence*, vol. 4, no. 4, 2001, 373-380.