Artificial Intelligence-Mediated Interaction in Virtual Reality Art

Jean-Luc Lugrin, Marc Cavazza, and Sean Crooks, *University of Teesside*

Mark Palmer, *University of the West of England*

A novel approach to the use of AI technologies in virtual reality art installations is based on the semantic representation of interaction events.

In entertainment applications, artificial intelligence techniques have most often been used to implement embodied agents or to automatically generate artistic content. A more recent development concerns using AI to support the user experience through new AI-based interactivity techniques. This is especially of interest for the development of artistic installations based on interactive 3D worlds.[1–4] A major difficulty in developing such installations is to properly translate the artistic intention into actual elements of interactivity, which in turn determine the user experience.

The starting point of this research was to facilitate the description of high-level behaviors for virtual worlds that would form part of virtual reality (VR) art installations. Our underlying hypothesis has been that AI representations inspired by planning formalisms,[5] and AI-based simulation derived from these, could constitute the basis for virtual-world behavior in these installations.

In our approach to interactivity, the consequences of user interaction can be dynamically computed to produce cascaded effects eliciting a specific kind of user experience. This chain of events is computed from first principles embedding elements of the artistic brief (the artist's initial conceptual description of the interactive installation and the intended user experience). In other words, AI techniques are used for their ability to represent actions and to compute analogical transformations on them to create a user experience.

System overview and architecture

Our system presents itself as an immersive virtual environment based on a CAVE-like device, the SAS Cube. The SAS cube was developed on top of the Unreal Tournament 2003 game engine, which serves as the main visualization engine and provides basic interaction mechanisms.[6] The Unreal engine has also been ported to immersive displays using the latest version of the CaveUT software, [5] which supports stereoscopic visualization as well as head and hand tracking.

In addition, what makes possible the use of AI techniques to simulate behavior in virtual worlds is the exploitation of a specific feature of game engines, namely the fact that they rely on event-based systems to represent all kinds of interaction. Event-based systems originated from the need to discretize physical interaction to simplify physical calculations: although the dynamics of moving objects would be subject to numerical simulation, certain physical interactions' consequences (for example, glass shattering following impact from a hard object) could be determined in a discretized system without having to perform complex mechanical simulations in real time. Our installation consists of a virtual world in which we can alter the normal laws of physical causality by substituting physical actions' default effects with new chains of events.

Our approach relies on the system's recognition of high-level actions from low-level physical events to generate semantic representations of the virtual world's events as they occur. In other words, from a low-level set of events such as collisions and contacts between objects, the system recognizes high-level actions (such as pushing, breaking, and tilting) that affect world objects, thereby attributing the same meaning to world events that the user would. These actions are represented in the system using a formalism that makes their consequences explicit. A real-time modification of this representation will alter an action's default consequences, thereby affecting the user's experience and sense of reality in the virtual environment.

In contrast to the game engine's native event-based system, which directly handles event calls at the object level, our system is based on a centralized event man-

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

agement module, called the *Causal Engine*. This module constitutes a generic layer between the game's physics engine and the objects interacting in the scene (see figure 1). The Causal Engine continuously intercepts event calls from the physics engine and replaces them with alternative events it computes.

Our alternative causality system is based on two main components communicating via UDP (User Datagram Protocol) sockets: an event-interception system and an action-modification module. The EIS is in charge of event interception, analysis, and reactivation, and the action-modification module computes action modification prior to the actions' reactivation (see figure 2). This mechanism generates artificial event co-occurrences in user real time, which, in turn, influence the user experience.

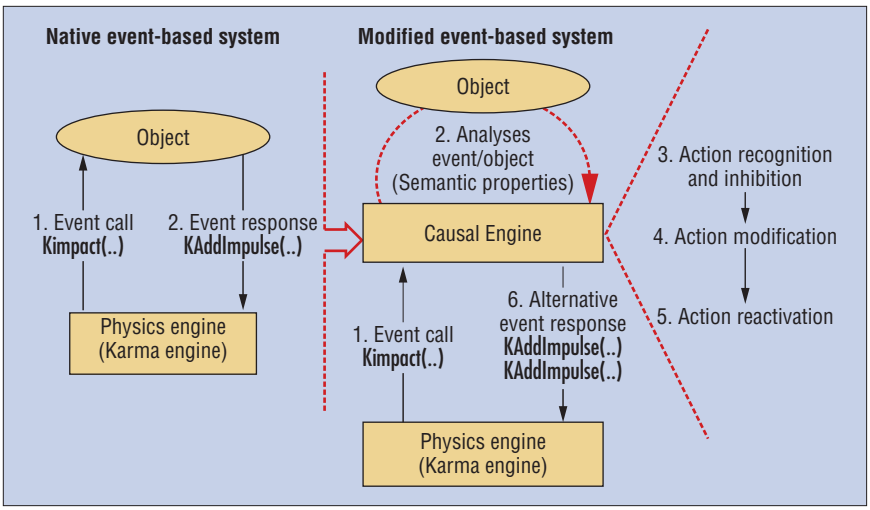

Figure 1. Integrating AI-based behavior in the game engine's event system.

Virtual reality art and alternative causality

VR art has been exploring complex user experiences in virtual environments. In that sense, a tradition exists in VR art of constructing alternative worlds—consider, for example, Char Davies' Osmose VR installation or Maurice Benayoun's Quarxs (featured in the eponymous animation series), invisible creatures that bend the rules of physics. However, the development of VR art installations to date has been dedicated mostly to the design of virtual worlds and ad hoc interaction mechanisms. One challenge for the further development of VR art would be to develop technologies supporting the definition of world behavior from first principles. These technologies would make new experimentations with more ambitious VR installations possible.

The technologies we discuss in this article have been used to support interaction in VR art installations. As an illustration, we present the first prototype of a VR installation one of us (Mark Palmer) created. Gyre and Gimble is a VR artistic project based upon Lewis Carroll's Alice stories. Although Gyre and Gimble draws upon the Alice stories, we intended not to reproduce their narratives but to explore the disruption to perception offered within them. In fact, Carroll's stories provide a natural starting point for anybody interested in the consequences of logic and the creation of alternative realities. The way that we often encounter this in his stories is through the mixing, collision, and invention of games as well as the transformation of their rules. Carroll's playfulness with language presents us with paradoxes arising out of the situations and conversations in which Alice finds herself. Take, for example, this conversation from *Alice's Adventures in Wonderland* and *Through the Looking Glass* (Penguin Books, 1998):

> "When we were little," the Mock Turtle went on at last, more calmly, though still sobbing now and then, "We went to school in the sea. The master was an old Turtle—we used to call him Tortoise—"
>
> "Why did you call him Tortoise, if he wasn't one?" Alice asked.
>
> "We called him Tortoise because he taught us," said the Mock Turtle angrily. "Really you are very dull!"

This is why his books are always far more than just the presentation of a plot; they're events that unfold, involving the reader in their own logic. Reacting to this interaction becomes the primary driver of Gyre and Gimble deliberately distancing itself from the narrative. Rather than becoming a vehicle for retelling portions of the Alice story, like the stories themselves, it becomes an "event" and an occasion that involves users in this disruptive process. The challenge was to make a technology based upon gaming that was as effective as Carroll's creative subversion of games. The joint decision to draw from the scene in *Through the Looking Glass* in which Alice discovers that, try as she might to look at things in a shop, they evade her gaze, was to provide the opportunity to use spectacle itself as a means of interaction. Using a combination of center of focus and the viewer's distance from the virtual objects under consideration let us employ attention as a means of interaction. The collision of objects that then occurred as a result of an object's desire to escape constituted, from the system perspective, the starting point for the computation of chains of events (and consequences).

The Gyre and Gimble brief's environment is a 3D world reflecting the aesthetics of Sir John Tenniel's original illustrations (that is, using 3D objects with nonphotorealistic rendering). The user, evolving in the environment as Alice in first-person mode, is a witness to various objects' behavior, which she can also affect by her presence. One room containing several cabinets and a table makes up the Gyre and Gimble world. It includes 10 different types of interactive objects, 90 in total, such as candles, candleholders, clocks, and books, dispersed on shelves, cabinets, and the table (see figure 3). The current version of the environment supports a set of 20 actions (such as tilting, breaking, flying, and burning).

User interaction

The user interacts with the VR art installation through navigation and by interacting with objects, as in any virtual environment. However, as part of the technical implementation of the artistic brief we described earlier, objects are associated a default behavior, by which they evade the user's gaze and escape toward other objects, depending on the object categories. The user's gaze triggers objects' spontaneous movement. Gaze is measured by the system from the head vector (with a fair approximation considering the average distance from the screens), and its direction is calculated using an approxi-

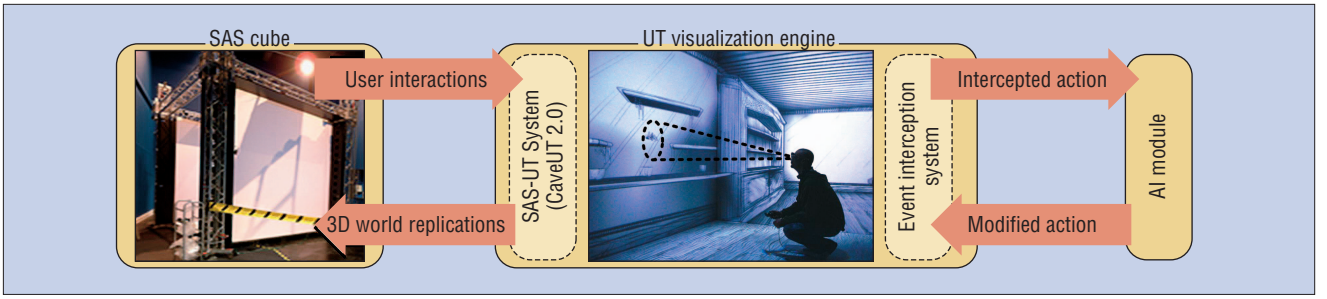

Figure 2. An overview of the system architecture. The system is based on a game engine ported to a CAVE-like immersive display.

Figure 3. The first version of the Gyre and Gimble environment and an example of interactive objects. Sir John Tenniel's original illustrations inspired the nonphotorealistic rendering.

mation from head-tracking data.

As figure 2 shows, a fixed cone of activation is associated with the user's field of view. Any object intersecting with the cone for more than a few seconds (randomly from two to 10) will start moving until it collides with another object. The object's motion depends on its type—some will slide, run, or jump away—and its speed depends on user-object distance (the closer they are together, the quicker the object moves). Simple contextual rules determine directions for escaping from the user's field of view. As a result, the user witnesses a stream of object behaviors, which interaction prompts but whose precise logic is not directly accessible to the user.

Another interaction mechanism, which is more specific to this type of installation, consists of integrating user trajectories and navigation patterns to determine the user's behavior toward specific parts of the environment and the objects they contain. Ultimately, those objects' behavior will reflect the user's behavior through global mechanisms involving degrees of perturbation and surprise, as the following sections describe.

Objects' spontaneous motion provokes collisions between them, which are the starting point from which the AI module generates an event chain. This chain of events will induce various forms of causal perception by the user and constitutes a central aspect of the interactive user experience. The amplitude of the alteration to causal event chains is based on semantic properties and analogies between objects and depends on how the user engages with the environment. This type of computation can provide a principled measure of concepts directly related to the user experience, such as "surprise."[7]

The virtual world ontology: Representing actions and objects

Knowledge representation is an essential aspect of our approach.[8] Our system is based on ontological representations for both objects and actions. These ontologies are developed specifically for the contents of a given virtual world, although significant subsets can be reused from one installation to another. The ontology for actions constitutes a specification of the main actions taking place in a given virtual world. The expression of an action's effects corresponds to its post-conditions (that is, a change in objects' properties). These effects are also associated with a visualization of the action itself in the form of a 3D animation of the action. In this way, the action can be properly staged in the virtual world without a detailed simulation of all its phases. This ontology contains both generic and specific actions. For instance, all objects can fall or fly, but only objects in certain categories can break (depending on their physical properties). In addition, some actions are related to object functionality (for example, only con-

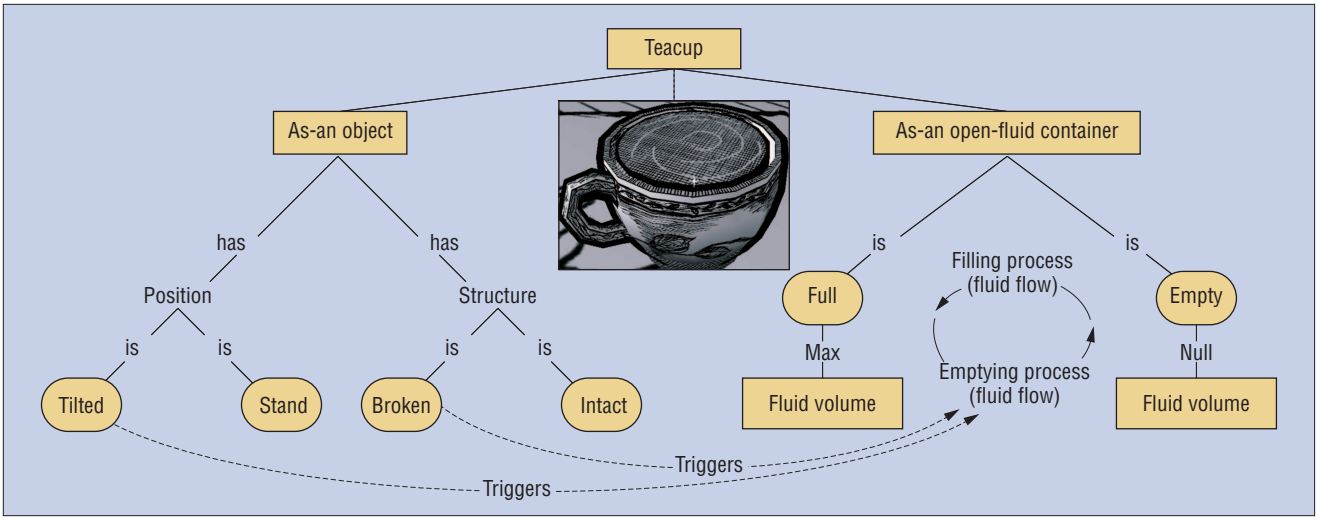

Figure 4. Structured representation of an object in the ontology.

tainers can be filled or emptied).

The ontology for objects serves different purposes:

- to determine which actions objects can take part in and how those actions will affect them, and
- to support comparison and analogies between various world objects.

The object ontology should support efficient computations to be compatible with a virtual environment's real-time requirements. Physical properties will appear in the preconditions, or triggers, of actions. For instance, an action such as **breaks-on-impact(?object, ?surface)** will test the surface's hardness. This will account for various situations, such as a cup breaking upon impact on a table or a window breaking when a ball hits it.

Analogical representations play an important role in generating chains of events that can induce causal impressions in the virtual world. The production of alternative consequences for actions is largely based on analogical transformations, such as substituting an effect for another or replacing the action's object with another one based on similarity criteria. We can draw analogies from physical or functional descriptions: for instance, the process of liquid evaporating from a container can be related to a candle burning.

Objects are represented in the ontology using a dual structure: one part is dedicated to the object's physical properties (such as size, shape, and hardness) and another to the object's functional properties (such as container or light source). Both parts include representations for predefined states, which are a key element of the representation (especially considering that the representation should support the derivation of action consequences). Predefined states correspond to an object's potential state according to its physical properties and functionality. Physical states describe high-level properties such as object orientation or the object's integrity. Functional states relate to the object's telicity: a candle can be lit or not, a container filled or empty, and so on. Transitions between states correspond to specific actions described as part of the ontology; the actions' postconditions correspond to the various object states. States can also relate the two descriptions: for example, the "tilted" positional state for a teacup would be compatible only with its "emptying" functional state as a fluid container (see Figure 4). Similarly, a book is readable (functional) only if it's open (physical). The connection between physical and functional states enables causal simulation without requiring complex qualitative simulation or commonsense reasoning to be implemented on top of the system.

Alternative reality and the generation of chains of events

The experience of alternative reality attached to artistic installations is based on the creation of chains of events responding to user interaction that induce impressions of causality. The two main principles for creating chains of events are the modification of certain actions' default consequences and the addition of new effects. Our system uses an ontology of actions, from whose representations it can generate alternative effects using the actions' semantic properties. Planning formalisms (namely the SIPE formalism[5]) inspired our action representations. They associate within the same representation an intervention in the form of a situated physical event (the cause) and its consequences in terms of object modifications (the effects). We've termed these representations CE (for cause-effect) structures. As we previously described, CE representations are continuously produced by "parsing" the low-level system events corresponding to physical contact between objects into CE structures, using the semantic properties of the objects taking part in those actions.

Figure 5 shows the CE representation for a projecting (that is, throwing) action **Tilt-Object(?obj1, ?obj2)**. Its "trigger" part corresponds to the physical event initiating the action **(Hit (?obj1, ?obj2))** and its "effect" part to the action's consequences (the fact that the object tilts upon impact). The "condition" field corresponds to physical properties characterizing objects taking part in that action (slender, light, and so on). These properties are part of the action's semantics and are necessary conditions to instantiate a CE.

An event chain's generation is based on the real-time transformation of CE instances while their effects are "frozen" by the EIS. From a formal perspective, these transformations should modify the contents of a CE instance to produce new types of events upon reactivation of the CE's effects. To do so, the Causal Engine relies on several features, which correspond to

- a semantics for the CE representation and

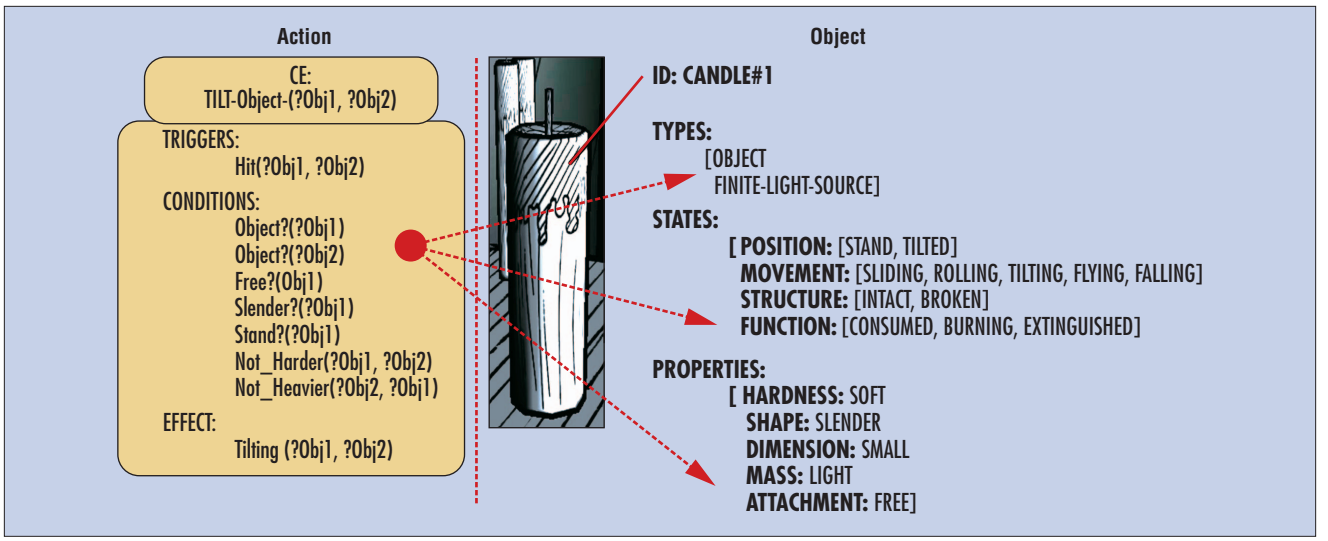

Figure 5. The cause-effect (CE) action and its use of objects' semantic properties.

the objects,
- specific procedures for transformation that alter the CE instance's contents, and
- a control mechanism for the set of transformations over the CE.

The first aspect derives from an underlying ontology in which objects are classified according to dimensions characterizing their physical properties (for example, soft, light, free, or solid). These semantic dimensions in turn determine some actions' applicability (as implemented in the "condition" part of a CE). For instance, only fragile objects can be the targets of shatter-on-impact CEs, and only objects of certain shapes can be affected by a tilt-object CE.

The second aspect is implemented through the notion of Macro-Operators (MOps)—specific procedures for performing expression substitutions within instantiated CE representations. MOps are described in terms of the transformation classes they operate on a CE's effects. Examples of MOp classes include

- change-object, which substitutes new objects for those that the CE originally affected;
- change-effect, which modifies a CE's effects (consequences);
- propagate-effect, which extends the CE's effects to other semantically compatible objects in the environment; and
- link-effect, which relates one CE's effect to another one's.

Finally, a control mechanism should determine which CE to modify and which modification is most appropriate from a semantic perspective. The control mechanism will select a set (or a sequence) of MOps to be applied on the candidate CE. The criteria for this selection must be semantic ones. As we'll describe in the next section, we based this control mechanism on a heuristic search algorithm.[9]

Event transformation

The CE creates chains of events through the principled modification of actions intercepted in real time. The basic principle for the dynamic creation of such event chains relies on degrees of analogy between default effects and alternative ones produced as part of event modification. These modifications are performed by applying a MOp to an intercepted event. The overall algorithm explores the space of potential transformations using heuristic search. It generates a set of possible transformations, from which it selects the most appropriate one in terms of semantic compatibility and spatial constraints. A global score measuring the level of disruption can control the search. This concept determines the nature of the user experience—that is, how much the virtual world departs from the everyday world.

The CE is the main module for creating causal chains from the transformation of intercepted actions. It operates through four steps: initialization, generation, evaluation, and selection.

Initialization

This first step consists of identifying a pool of compatible objects to which effects could be propagated to create a chain of events. These are initially selected on the basis of their spatial properties—that is, proximity to the user or his or her field of vision. This pool determines the set of objects that the MOp will manipulate to serve as substitute objects for the CE effect or to which the CE effect will be propagated. When the initialization process is completed, each action representation will have been altered independently. However, to avoid conflicts between modifications, the algorithm removes from the pool objects already involved in an intercepted event.

Generation

Alternative effects are generated by successively applying a series of MOps on a "frozen" CE. The list of MOp classes the Causal Engine uses is determined offline. Here, the Causal Engine will successively execute two MOps: CHANGE-EFFECT and PROPAGATE-EFFECT. Typically, a MOp application creates a new instance of the frozen CE and modifies its EFFECT section. In our system, an EFFECT is identified by a type, a state, and an object's reference on which the effect will be applied (see following example). In our object representation, we define four types of effects depending on the object properties they alter: POSITION, MOVEMENT, STRUCTURE, and FUNCTION. An effect is represented by a tuple:

EFFECT [MOVEMENT, TILTING, Candle#1]

A MOp will change either the EFFECT tuple's object (Candle#1) or its effect field (TILTING).

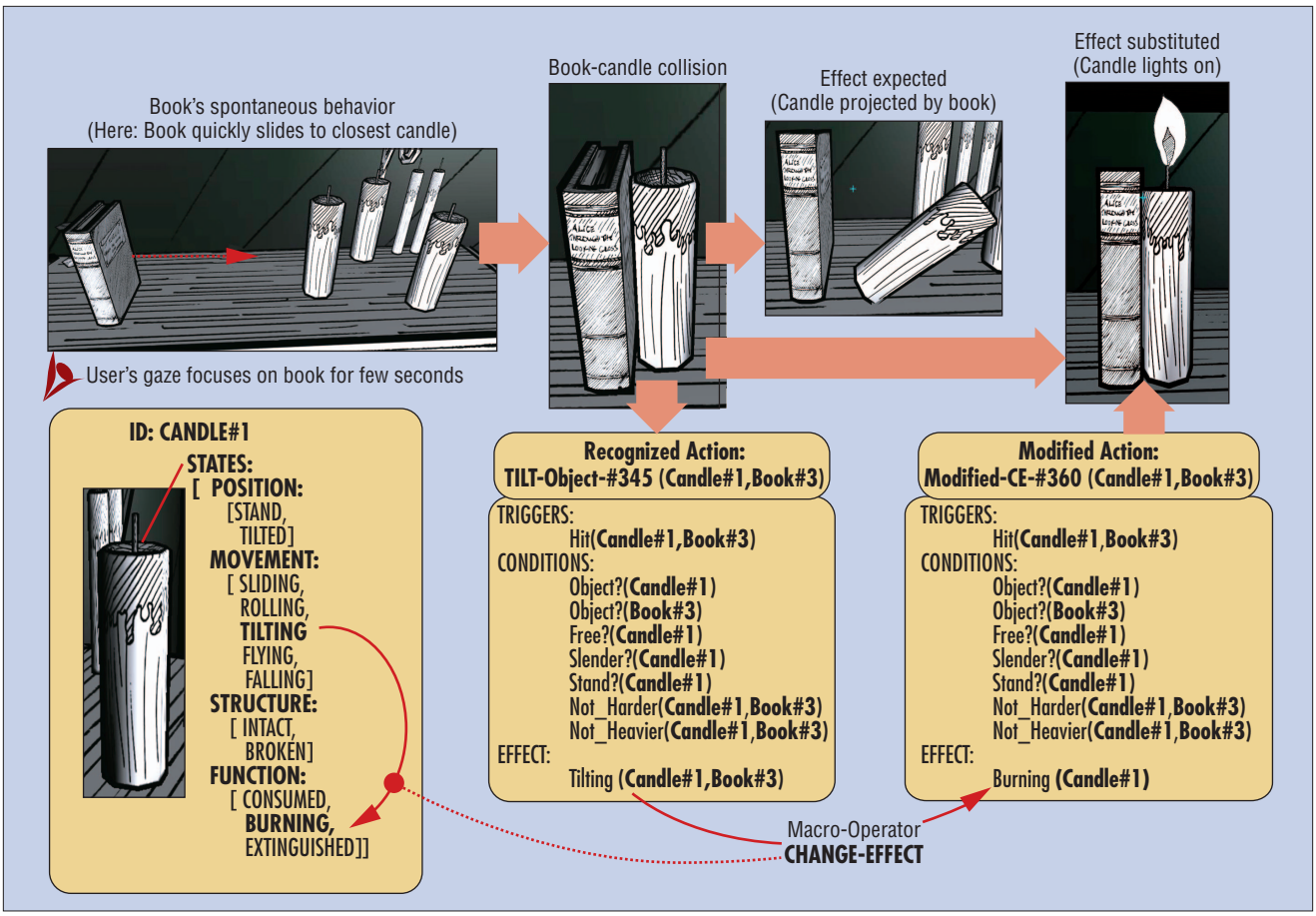

Figure 6. The application of the CHANGE-EFFECT Macro-Operator (MOp). The book hits the candle while trying to escape the user's gaze. Upon impact, the candle should normally tilt forward. However, the system intercepts the action and modifies its default consequences using MOp, resulting in the candle being set alight instead.

MOps access an object's states to determine which effects the object supports (see figure 6). For each potential state or for each object involved in the effect, a MOp creates a possible modification. For instance, a CHANGE-EFFECT MOp will produce 10 potential modifications if an object supports 10 different states (as our candle object does). Similarly, a PROPAGATE-EFFECT MOp produces a new potential modification for each object within the candidate pool (providing these support the particular effect propagated). Consequently, if four candles (including the one colliding) are in the user's field of view, the list of 10 modifications produced by the previous MOp will be expanded to reach 40 candidate modifications.

From the set of candidate modifications, further MOps can be applied to each modification. This process of successive MOp applications generates a modification space. The constraints on number of object states and number of compatible objects determine the modification space's branching factor at each application step. The generation process is terminated once all relevant MOps have been applied or the process has exceeded 50 milliseconds. Considering that the number of modifications can increase exponentially, this time threshold preserves a response time compatible with a virtual environment's requirements.

At the end of the generation process, the system has generated a set of candidate modifications applicable to the "frozen" CE (these modifications take the form of a partial path in the search space). To maximize the chance of causal attribution by the user, these candidate modifications should be evaluated for consistency and relevance in the context of the original CE.

Evaluation

Each modified CE is associated a value reflecting its degree of similarity with the initial CE (measuring the extent of the transformation). For modifications involving objects other than the default one, their spatial distribution score is also computed. These scores are aggregated into a degree of plausibility for each transformation, represented by a value normalized between zero and one. The search algorithm uses this value as a heuristic function to determine the most appropriate transformation. Each modification can be associated with a triple:

MODIFICATION [[MOVEMENT, TILTING, Candle#1], [FUNCTION, BURNING, Candle#1],0.8]

The modification cost behaves as a heuristic value reflecting the level of change the modification introduced. This cost is determined by means of a "heuristic" comparison between the original and the modified effects, reinforced by spatial considerations between

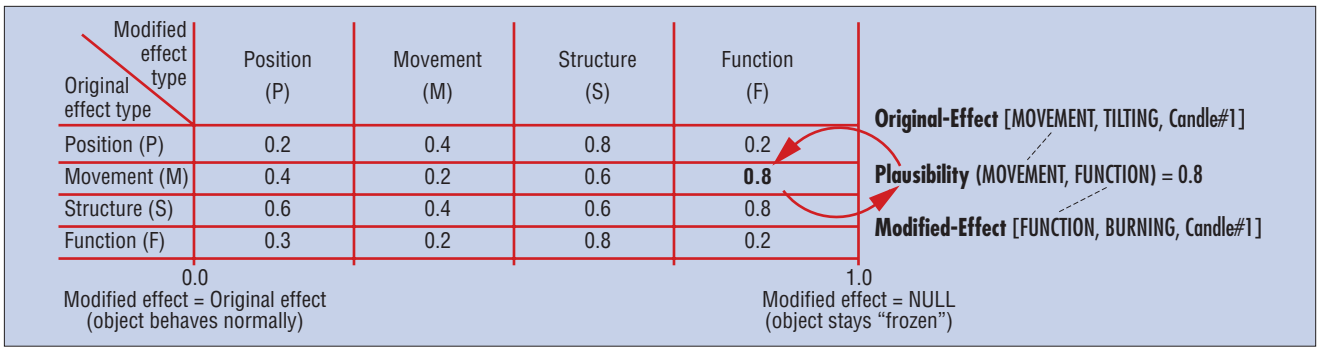

Figure 7. The Plausibility matrix provides heuristic coefficients based on the difference in type between the original and modified effects.

the original object position and that of the new object involved in its modified version:

Modification Cost =
Plausibility Weight * Proximity Weight

The plausibility weight corresponds to a modification's degree of plausibility (which in our case is equivalent to the extent of change). A function computes this weight using a simple matrix associating a heuristic value to each possible combination (see figure 7). The values in this figure range from zero to one, zero indicating that the modified effect the MOp produced is equal to the original effect. A value of one expresses a total absence of effects, which is considered the less natural consequence. For instance, a candle projecting against a wall will remain "stuck" to it in its position at impact. The smaller the value, the more plausible (or less disruptive) the transformation is. For instance, changing a MOVEMENT effect such as Tilting with another MOVEMENT effect such as Sliding appears much less disruptive than replacing Tilting with a FUNCTION-type effect such as Burning. The plausibility matrix was initially established by identifying analogies between potential consequences of a sample set of actions. In a subsequent step, we readjusted the weights associated with the matrix elements according to feedback from user experiments, as we explain in the next section.

However, the spatial contiguity of co-occurring events also considerably influences a modification's plausibility. Because distant events are less likely to be perceived as correlated, we've complemented the plausibility weight by a proximity weight accounting for the influence of events' locations. A proximity weight valued at 0 corresponds to the original object; a value of 1 represents the farthest object.

Selection

During this last step, the algorithm uses the heuristic values computed in the previous step to select the modification closest to a predefined cost threshold. This threshold represents the "global" modification cost the Causal Engine targets. We refer to this value, normalized between 0 and 1, as the global *level of disruption*. For instance, if we set the level of disruption at 0.8, the algorithm will select the modification presenting a cost around 0.8 and then reactivate it in the environment instead of the normal consequences (for example, a candle will start burning instead of tilting). In other words, the level of disruption can be considered as a heuristic threshold used to determine the extent of transformations, hence affecting the user experience.

The combination of MOps and heuristic search provides an original approach that allows a systematic exploration of a transformation space. Overall, the system performance is in line with its initial design constraints, which imposed a response time in the order of 60–120 ms[10] (with a frequency of 20–50 intercepted events per minute). The event recognition and reactivation process is achieved between 40 and 60 ms, while the event modification occurs in a time range of 20–40 ms.

Experiments

To evaluate the system's capacity to elicit causal perception, we conducted two experiments involving a total of 60 subjects. For both experiments, we tested the system in a desktop configuration. We instructed subjects to repeat a similar action while our system exposed them to different consequences every time. For instance, in our first experiment, users had to drop a glass on a table supporting several objects. The default effect was for the glass to shatter on impact. The system generated alternative co-occurrences, such as another glass on the table shattering, the table surface cracking, another glass tilting over and spilling its contents on the table, or a cardboard menu falling from the table.

For each experiment, we instructed the subjects to explain the events immediately after each try. We analyzed transcripts of these explanations for the occurrence of causal vocabulary and explanations. These experiments showed that the subjects perceived the effect substitution as a causal relation in more than 70 percent of individual tasks. This is comparable to scores previously reported in psychological experiments dealing with causal perception.[11] The experiments also led us to adjust and validate our heuristic search. However, our current work consists of refining our plausibility heuristic by introducing metrics to measure action analogy based on condition and triggers similarity.

System response and user experience

As we explained earlier, the level of disruption corresponds to the search threshold for the use of heuristic values. According to this threshold's value, the transformation produced goes from the more natural (0) to the more artificial (1). Our system discretizes this value into five disruption levels: NULL, LOW, MEDIUM, HIGH, and VERY-HIGH. In the Gyre and Gimble world, the system dynamically updates the level of disruption to represent the environment's response to the user's perceived behavior toward it. Two regularly updated parameters represent user behavior: (User-Objects-Proximity) integrates the amount of time the user spent in proximity with certain objects, and (User-Activity) reflects the user's exploration of the world. More specifically, (User-Object-Proximity) is weighted between 0 and 1 and corresponds to the user's

average distance from the objects he or she has activated. This metric reflects the user's level of engagement with an object. Depending on the artistic brief, this can be interpreted as interest or a threat. (User-Activity) represents an appreciation of the user's frequency of movement and interaction with the world. Every 10 to 30 seconds, the distance the user covers and the number of objects he or she has activated are compiled into a score between 0 and 1, a value of 0 meaning that the user has been nearly immobile.

The level of disruption is then frequently updated using a simple matrix (see figure 8). Increasing or decreasing it in response to user behavior creates different user experiences in terms of emotions reflected in, and by, the world itself. The user thus indirectly influences the transformation amplitude through values for his or her behavior.

This constitutes another example of the generation of more sophisticated user experience through AI technologies. In other artistic installations, the level of disruption is influenced by metrics involving the user's degree of empathy toward artificial creatures he or she can decide to repulse or attract.[1]

A low value for the level-of-disruption parameter (close to 0.25) tends to result in minor changes. Indeed, they're often related to the propagation of a normal consequence to spatially close or same-type objects. For instance, the book-candle collision will also project (that is, throw) one of the closest similar candles. However, a medium level of disruption (around 0.5) usually extends or substitutes default consequences to different objects, as when the book is projected with the candles around, instead of the original candle (see figure 9, number 2: Action modification). Higher levels of disruption (close to 1.0) affect the type of effects generated and the entire population of objects in the user's field of view. At this level, an interaction's consequence becomes hardly predictable because it depends on the environmental context (that is, the type, state, and distance of the objects surrounding the initial event). Here, such a level triggers the opening of the book while some candles start burning or tilting (see figure 9, number 3: Actions reactivation).

This approach's advantage is its ability to control the consequences of user interaction at different levels, using concepts that can be related to artistic intentions. Most importantly, these principles also support generative aspects, where the system enriches the creative process.

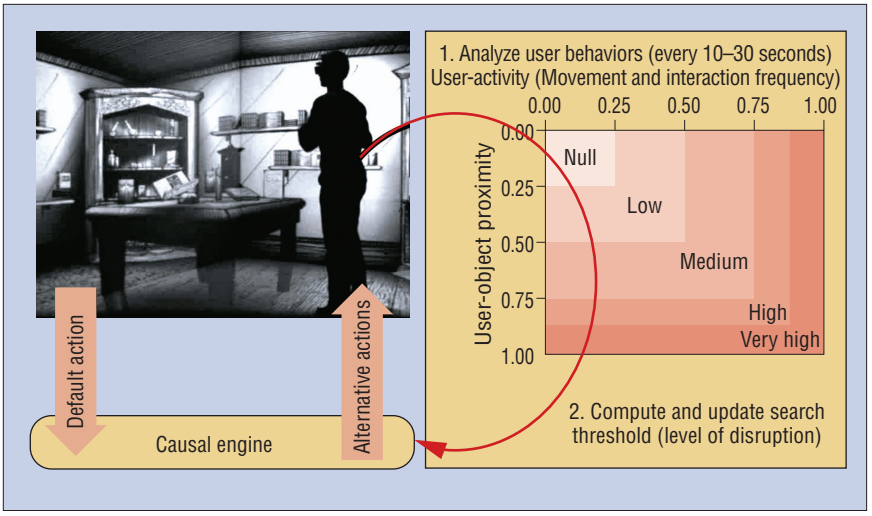

Figure 8. The user behavior is interpreted in terms of her relation to the environment and determines the environment reaction.

We've applied our approach to two fully implemented artistic installations from two different artists.[1,2] We carried out evaluation sessions with the artists to

- obtain the artists' feedback on how the final installation implemented their early ideas about interactivity as stated in the original briefs and
- enable them to experiment with the system formalisms to fine-tune the installation's behavior.

These sessions helped us focus our future work into two directions. The first one quite naturally consists of granting artists better access to the system's formalisms through the development of specific authoring tools. The second is to investigate more systematically users' responses to these installations. Recent research in entertainment theory suggests a possible approach for such evaluation.

Traditional interactive systems rely on direct associations between interaction events and their scripted consequences. This has some limitations for digital-arts applications, forcing the specification of all low-level events when implementing an artistic brief. Such an approach also has limited flexibility when it comes to eliciting more complex user experience, such as reacting to the user's global behavior. An AI perspective brings two major advances. The first is an explicit representation layer for high-level actions, which supports principled modifications of actions' consequences. These principles are derived from the high-level specification of system behavior produced as part of the artistic briefs. For instance, the Gyre and Gimble storyboard refers explicitly to causality and describes various levels of disruption to the environment's expected behavior. The other advantage is that the AI approach's generative properties enrich the user experience, simplifying the authoring process. As VR art develops,[3] the requirements on advanced interactivity will become more demanding. Mediating interaction through AI representations seems a promising research direction, as several installations that we've been supporting suggest.[1,2] ⬒

Acknowledgments

This research was funded in part by the European Commission through the ALTERNE (IST-38575) project. We thank Marc Le Renard and Jeffrey Jacobson for their participation in the SAS Cube and CaveUT developments.

References

1. M. Cavazza et al., "Causality and Virtual Reality Art," *Proc. 5th Conf. Creativity and Cognition*, ACM Press, 2005, pp. 4–12.

2. M. Cavazza et al., "New Ways of Worldmaking: The Alterne Platform for VR Art," *Proc. 12th Ann. ACM Int'l Conf. Multimedia*, ACM Press, 2004, pp. 80–87.

3. O. Grau, *Virtual Art: From Illusion to Immersion*, MIT Press, 2002.

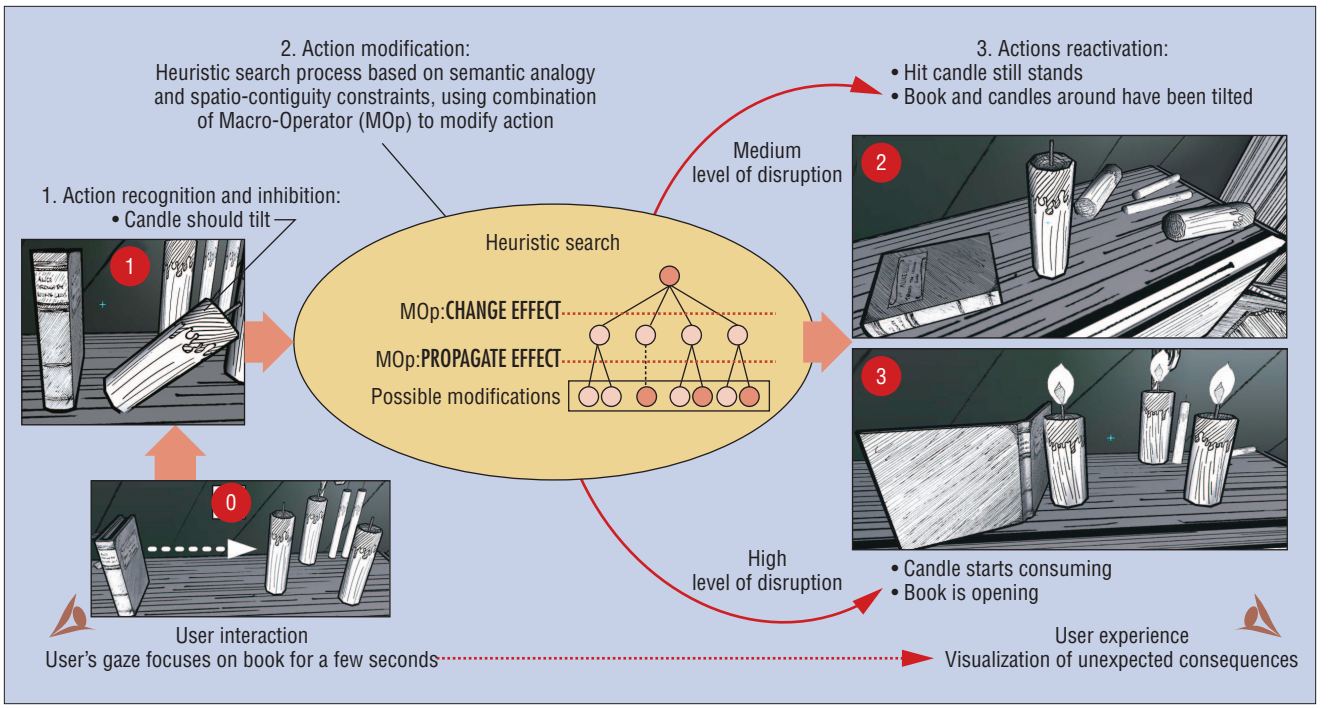

Figure 9. The "level of disruption" is a parameter governing the generation of unrealistic causality.

4. J. Jacobson et al., "The CaveUT System: Immersive Entertainment Based on a Game Engine," *Proc. 2nd ACM SIGCHI Conf. Advances in Computer Entertainment Technology* (ACE 05), ACM Press, 2005, pp. 184–187.

5. D.E. Wilkins, "Causal Reasoning in Planning," *Computational Intelligence*, vol. 4, no. 4, 1988, pp. 373–380.

6. M. Lewis and J. Jacobson, "Game Engines in Scientific Research," *Comm. ACM*, vol. 45, no. 1, 2002, pp. 27–31.

7. L. Macedo and A. Cardoso, "Modelling Forms of Surprise in an Artificial Agent," *Proc. 23rd Ann. Conf. Cognitive Science Society*, Lawrence Erlbaum Associates, 2001, pp. 588–593.

8. J.M. Zacks and B. Tversky, "Event Structure in Perception and Conception," *Psychological Bull.*, vol. 127, no. 1, 2001, pp. 3–21.

9. B. Bonet and H. Geffner, "Planning as Heuristic Search," *Artificial Intelligence*, vol. 129, nos. 1–2, 2001, pp. 5–33.

10. B.J. Scholl and P. Tremoulet, "Perceptual Causality and Animacy," *Trends in Cognitive Sciences*, vol. 4, no. 8, 2000, pp. 299–309.

11. A. Michotte, *The Perception of Causality*, Basic Books, 1963.

The Authors

Jean-Luc Lugrin is a lecturer in games programming at the University of Teesside. His research focuses on knowledge representation and new behavioral approaches for virtual reality as well as immersive displays. He's working on AI-based world behavior for emergent narratives. He received his MSc in computer graphics from the University of Teesside. Contact him at the Univ. of Teesside, School of Computing, Borough Rd., Middlesbrough TS1 3BA, UK; j-l.lugrin@tees.ac.uk.

Marc Cavazza is a professor of intelligent virtual environments at the University of Teesside. His research interests are in the application of artificial intelligence techniques to virtual reality. He has been coordinator of the EU-funded ALTERNE project, of which this research is part. He received his PhD in biomathematics from the University of Paris 7. Contact him at the Univ. of Teesside, School of Computing, Borough Rd., Middlesbrough TS1 3BA, UK; m.o.cavazza@tees.ac.uk.

Marc Palmer is a senior lecturer in games technology at the University of the West of England. His research interests are in the creation of genuine emergent multiuser systems where notions of "artist" and "user" become blurred. He received his PhD at Staffordshire University. He's the chair of Computers in Art and Design Education. Contact him at the School of Computer Science, Faculty of Computing, Eng. and Mathematical Sciences, Univ. of the West of England, Bristol, Frenchay Campus, Cold Harbour Lane, Bristol BS16 1QY, UK; mark.palmer@uwe.ac.uk.

Sean Crooks is a research assistant and creative director of a computer games development studio based at the University of Teesside. He's researching the use of virtual environments to simulate the spread of obesity while handling the production of various computer game titles. He received his BSc in virtual reality from the University of Teesside. Contact him at the Univ. of Teesside, School of Computing, Borough Rd., Middlesbrough TS1 3BA, UK; s.crooks@tees.ac.uk.