New Behavioural Approaches for Virtual Environments

Marc Cavazza[1], Simon Hartley[1], Jean-Luc Lugrin[1], Paolo Libardi[2], and Mikael Le Bras[1]

[1] School of Computing, University of Teesside, TS1 3BA, Middlesbrough, United Kingdom
m.o.cavazza@tees.ac.uk
[2] Department of Electronics for Automation, University of Brescia, via Branze 38, Brescia, I-25123, Italy

Abstract. We describe a new approach to the behaviour of 3D environments that supports the definition of physical processes and interactive phenomena. The work takes as a starting point the traditional event-based architecture that underlies most game engines. These systems discretise the environments' Physics by separating the objects' kinematics from the physical processes corresponding to objects interactions. This property has been used to insert a new behavioural layer, which implements AI-based simulation techniques. We introduce the rationale behind AI-based simulation and the techniques we use for qualitative Physics, as well as a new approach to world behaviour based on the induction of causal impressions. This is illustrated through several examples on a test environment. This approach has implications for the definition of complex world behaviour or non-standard physics, as required in creative applications.

1 Introduction

It is a common view in interactive systems research to consider that, while graphics and visualisation have made significant progress over recent years, behavioural aspects are somehow lagging behind, and have not sustained a similar pace of progression. This generic statement concerns both the simulation of realistic Physics for 3D worlds and the behaviour (generally AI-based) of autonomous entities populating them.

One well-known instantiation of this statement consists in saying that the added value of future computer games will derive increasingly from the AI technology they incorporate, although this statement probably needs to be revisited from a more fundamental perspective. More realistic physical modelling also constitutes a challenge, in particular in terms of computational resources. This problem is currently approached by discretising physical simulation to reflect the actual events taking place in the virtual world, in particular those arising from interaction between world entities.

There has been, generally speaking, little research on the notion of integrated world behaviour, which goes beyond the isolated simulation of object's physical behaviour

to consider the world's physical phenomena from a more global perspective. Such an approach would be centred not only on physical objects but also on processes affecting objects, on the aggregation of objects into systems or devices whose behaviour cannot be deduced directly from simple physical simulation, but requires a higher level of conceptual modelling. It should also consider relations between events occurring in the 3D world, and how these can be perceived as causally related by the user.

In this paper, we introduce a novel approach to the integrated behaviour of virtual worlds, which is based on the use of AI techniques derived, among others, from qualitative simulation. We first describe the rationale for this approach and the system architecture, which is organised around the Unreal Tournament 2003™ game engine [1], whose event-based system serves as a baseline layer for integration of high-level behaviour. We then discuss two novel methods supporting 3D world's behaviour, which are qualitative physics and causal propagation, and how their basic components and formalism have been adapted to the specific constraints of real-time visualisation.

1.1 Physics Modelling, Event-based Systems and Interaction

Comprehensive modelling of all physical events in a virtual environment would be a formidable task, impossible to achieve in real time. In interactive virtual environments, basic physical behaviour is implemented in an interactive fashion to maintain a response rate which is acceptable for user interaction. This has led to the rationale, that in order to maintain this interaction rate, the Physical simulation is discretised. Kinematic aspects within these systems tend to be simulated through traditional numerical approaches, while more complex mechanical events (objects breaking or exploding) are pre-calculated. In other words, the overall dynamics of objects is subject to traditional physical simulation, while interactions between objects (collisions, etc.) constitute discretisation units. This saves considerable computation at the level of these events, whose pre-computed consequences can be triggered as a consequence of event recognition. The condition for such a system to work efficiently is the availability of an "event system", i.e. mechanisms for recognising in real-time the occurrence of such events and supporting the programming of cause-effects associations. In most cases, event systems are derived from basic collision detection mechanisms of the graphic engines. These systems are able to produce event primitives corresponding to collision or contact between objects, or objects entering and leaving areas or volumes in the 3D environment.

This is not specific to game engines, as event-based systems play an important role in VR software as well [2]. However, traditional game engines use their event system essentially as an API supporting the ad hoc development of object's behaviours associated with specific instances of events. The starting point for this research was to use the inherent discretisation of Physics in 3D engines to integrate high level behavioural mechanisms that could support complex behavioural simulation on a principled basis.

1.2 System Architecture

The system comprises a graphic environment, composed of the UT 2003 engine and an external physical simulation modules (called QP engine and causal engine, see below), developed in C++. The software architecture is based on UDP communication, supported through the UDPLink class in UT 2003. The messages exchanged between the UT 2003 environment and the behavioural modules correspond, on one side, to the activation conditions of various behaviour instances run by the engine. On the other side, the engine sends messages to update object states, which are interpreted by the Unreal Environment [3].
The Unreal Tournament engine extensively relies on event generation to support many of its interaction aspects and, most importantly, the mechanism for event generation is accessible to redefine specific behaviours. Formally, an event can be characterised as an encapsulated message, which is generated by an Event Source, this being an object of the environment. Examples of such basic events are: `Bump(Actor Other)`, `Touch(Actor Other)`, `UnTouch(Actor Other)`, `ActorEnteredVolume (Actor Volume)`, etc.
The Unreal Tournament Engine implements two different kinds of event: the basic (primitive) events, which are low level events defined within the game engine (derived from the collision detection procedures in the graphic engine), and the programmed events. The latter are events whose definitions are scripted and can thus be programmed by the system developer. This is a mechanism by which the system can parse low-level events into high-level events corresponding to a semantic description of the world.

1.3 Techniques for World Behaviour

The notion of *world behaviour* generalises that of virtual world Physics, to encompass any kind of dynamic generation of events in the virtual world. In traditional physical simulation, pre-defined consequences are triggered in response to specific events, such as a glass exploding when hit by a missile. This approach can be generalised by considering the principles according to which events can be related to one another. For instance, physical simulation can be entirely discretised using laws of physics to produce causal chains of elementary events. This approach to symbolic reasoning on physical systems corresponds to an AI technique known as Qualitative Physics [3]. On the other hand, it is also possible to produce world behaviours by directly relating events to one another so as to create artificial causal chains. We describe both methods in the remainder of this paper. Like with any AI method, but more specifically with those modelling worlds, we should first discuss the knowledge representation formalisms that support their implementation.

Process: Fluid-flow (?source?sub ?dst ?path)

Individuals: ?source a contained liquid
?destination a contained liquid
?sub a substance
?path a fluid-path

Preconditions: Connects(?path,?source,?dst)
Aligned(?path)

Quantity Conditions:
A[Pressure(C-S(?sub, liquid, ? source))]
> A[Pressure(?dst)]

Relations:
Quantity(flow-rate) Flow-rate=Pressure(C-S(?sub,liquid,source)) -Pressure(?dst)

Influences:
I+(Amount-of-in(?sub, liquid, ?Source), A[flow-rate])
I-(Amount-of-in(?sub, liquid, ?dst), A[flow-rate])

Figure 1. An Example Qualitative Process describing Fluid Flows. This process is interactively triggered by moving the glass under to running water.

2 Ontology and Representations

The principled definition of behaviour in a symbolic system relies on the appropriate description of action and processes, as well as object properties, which are determinants for their involvement in certain classes of actions and processes. It is thus necessary to develop an ontology for a given environment's Physics. Importantly, this ontology will describe both objects and relevant actions in the environment.

2.1 Representing Actions and Processes

The default mechanism for representing actions UT 2003™, which is representative of large class of interactive 3D system, consists in directly associating physical events to a set of possible consequences depending on the objects involved. For instance, the impact of a fragile object on a hard surface will be associated with this object being broken, through specific, ad hoc, scripting.
This description should be supported by an appropriate formalism for change-inducing events, which should clearly identify actions and their consequences. The second step consists in defining an ontology of such events, i.e. describing the most important high-level events that can be recognised in the virtual environment.
We have termed these change-inducing events Context Events (CE) to reflect their semantic nature. Typically, a CE is represented using an action formalism inspired from those serving similar functions in planning and robotics, such as STRIPS [4] or the operator representation in the SIPE system [5].

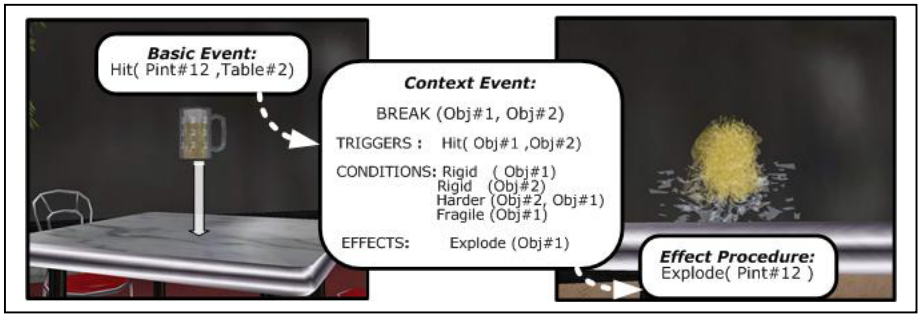

Figure 2. Context Events constitute high-level descriptions of actions in the virtual environment. Here, a fragile object breaking on impact (instantiated to a falling glass).

These representations originally describe operators responsible for transforming state of affairs in the world. They tend to be organised around pre-conditions, i.e. conditions that should be satisfied for them to take place and post-conditions, i.e. those world changes induced by their application.

Our formalism for CE comprises three main fields, which are analogue to the SIPE representation. The first field, called trigger, contains the basic event from which the CE can be recognised and which prompts instantiation of the corresponding CE. The condition field is a formula testing properties of the objects involved in such as CE. The effect field corresponds to the consequence part of the CE and contains the effect to be applied to the objects affected by the CE (Figure 2).

Another kind of action representation consists of discretised physical processes such as those used by qualitative physics. These Qualitative Processes (QP) encapsulate the expression of physical laws relevant to a given high-level process, e.g. liquid flows, heat transfer, etc. An example qualitative process is shown on Figure 1. It formalises several relevant aspects, from the conditions that trigger the activation of a process to the formulas which determine the evolution of key variables (influence equations). The set of formalised qualitative processes constitute an ontology of physical transformations for a given virtual world.

2.2 Objects' Representation

Objects descriptions vary greatly depending on the kind of processing which is applied to them. Often, part-whole relationships and functional properties tend to dominate symbolic object descriptions. In our specific context, the main role of object descriptions is to determine the kind of actions and processes they can take part in, as well as relating an object's visual appearance to the transformations that can be applied to it. In the first instance, we have organised these representations according to several dimensions: i) the object's mechanical properties (e.g. `breakable`, `movable`, etc.), ii) its functional properties (e.g. object as `container`, `fluid`

`source`, `support`, etc.) and iii) its visual properties (includes the object's appearance, but also the visual translation of certain of its behaviours, for instance for a glass to tilt, etc.). We have made a choice for the overall granularity of the representation, where direct relations can map properties of a given field onto another. This, in order to avoid the computational overhead of managing a complex semantic network .

Another aspect of object representation consists in relating them with the kind of qualitative processes they can take part in. The integration of Qualitative Physics in the virtual environments' basic mechanisms is achieved through the redefinition of a special class of physical objects: qualitative process objects, or QP objects. This follows traditional implementation techniques by which classes of objects are defined depending on the computations they can trigger from the interactions they participate in (i.e., in UT 2003, objects manipulated by the native physics engine, Karma™, are defined as members of the class of "Karma™ objects"). The QP objects have several properties: i) they are associated with an event interception mechanism that attaches data-driven procedures for the recognition of the pre-conditions of QPs in which they can take part, ii) their properties can be defined through qualitative variables, which are the key variables defined within QP's and are involved in qualitative proportionalities and influence equations, iii) they are associated physical states that have a visual translation, including in terms of transitions between states (e.g. animations showing a recipient filling, a liquid evaporating, etc.). These states correspond to landmark values for the qualitative variables.

3 Qualitative Physics in Virtual Environments

Of the various approaches that have been described in qualitative physics, we have opted for Qualitative Process Theory (henceforth QPT) [6], essentially for its representational properties. QPT descriptions are centred on physical processes (e.g. liquid flows, heat transfer, etc.) whose states are described through the values of qualitative variables. We have given a brief introduction to this formalism in previous sections. In essence, relations between variables are described through influence equations and qualitative proportionalities. The former correspond to the actual dynamics of the process; for instance, that the amount of liquid in a recipient increases with the inflow. The latter maintain "static" relationships between variables, such as the fact that the mass of liquid in the container is proportional to its volume. The QPT formalism is well adapted to its integration in virtual environments, for several reasons: i) the explicit description of a QP's pre-conditions supports the definition of procedures activating the QP simulation from physical events involving objects in the virtual world. This is the basic mechanism for integration of QP's in the interactive environment, ii) The kind of causality associated with QP descriptions can be matched to user interventions, and iii) QPT has been successfully used to define ontologies with a significant number of processes, representing a whole subset of physical processes for a given world.

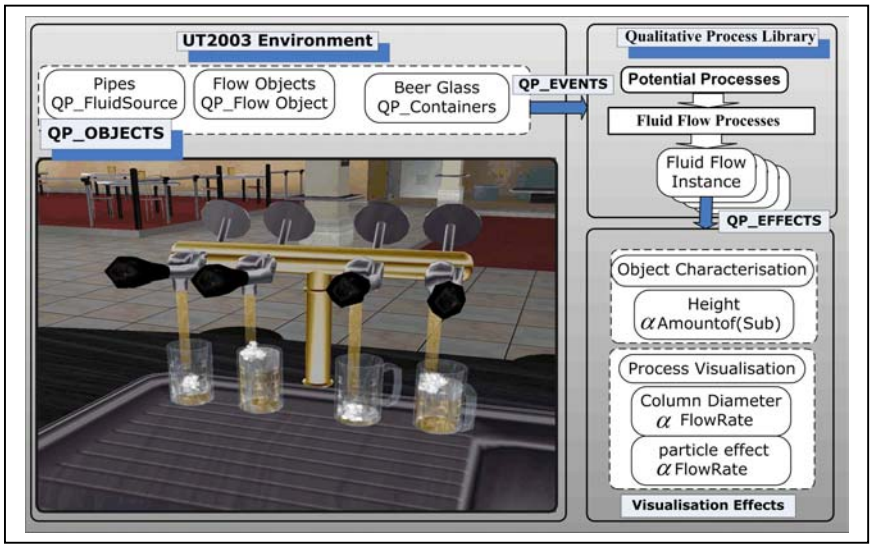

Figure 3. Several Qualitative Processes operating simultaneously can be visualised in the virtual world.

In terms of their actual implementation, pre-conditions are encoded in specific UnrealScript™ (the programming language of UT2003 serving as an API) procedures associated to the virtual world objects' in order to trigger the activation of relevant QPs. In that sense, pre-conditions are not strictly speaking part of the actual QP representation implemented.

However, all the other elements of the QP representations; qualitative variables, qualitative proportionalities and influence equations are implemented within the QP engine. Their actual use by the engine during simulations is discussed in the following sections.

Figure 3 shows the behaviour of the system simulating the filling of a glass from a running tap. When objects, which can behave as recipients, are aligned with a liquid flow (here the beer tap), this, corresponding to the pre-condition of a filling process, activates the corresponding liquid-flow QP on these objects. The process running in the QP engine updates the value of the amount of water in the glass, through its influence equations. In a similar fashion, qualitative proportionalities update the total mass of the glass, as well as the height of liquid. These variables transform the state of the filling glass in the virtual world by updating its physical properties (e.g. weight) as well as its appearance. The overall dynamics is dictated by the QP simulation process, the speed of the filling glass animation being an approximation of these dynamics. The overall simulation remains interactive, and at any time, the user can remove the glass from the running tap, which will interrupt the process while retaining the physical properties of the glass (amount of water/beer filled into the glass).

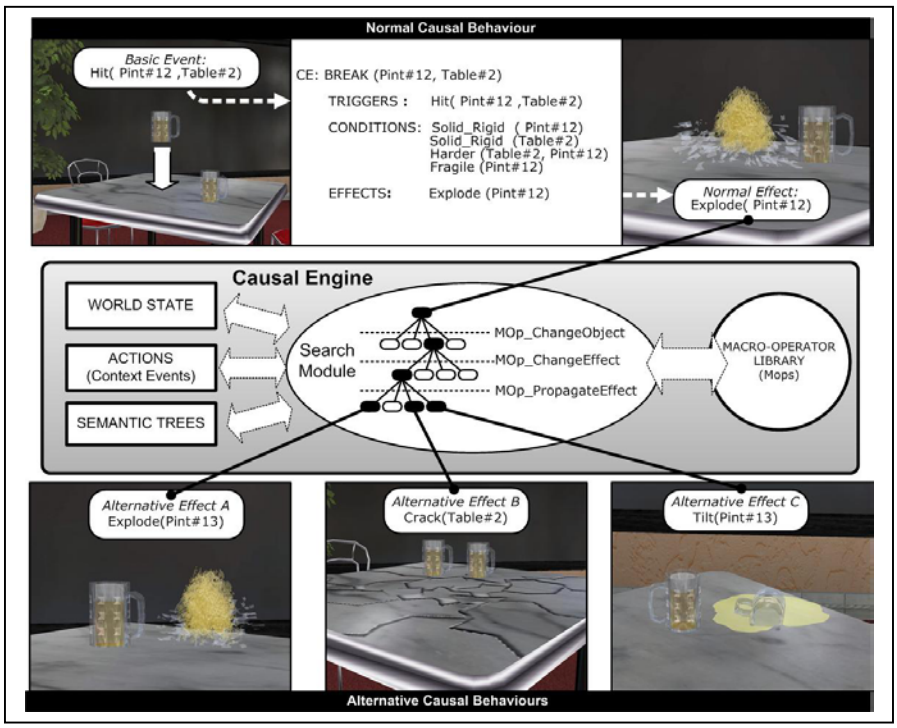

Figure 4. The Causal Engine can dynamically generate new consequences for events taking place in the virtual world by inhibiting the default outcome and substituting it with alternative effects. It relies on semantic descriptions of the actions and objects involved.

A typical description of world behaviour contains many QP that can interact dynamically, so as to reflect the behaviour of complex systems or devices that would be difficult to model in an integrated fashion through numerical simulation.

4 Redefining the Laws of Causality

Causal Simulation normally relates one event occurring in the virtual world to its logical consequences. Causal simulation is a technique for generating behaviours using symbolic descriptions of causal chains relating physical events.
In that sense, qualitative physics, as introduced above, incorporates causality within the symbolic description of physical processes and the physical laws governing them.

However, if we consider the behaviour of a virtual world as the one *perceived* by human users rather than an absolute one, it should be characterised by the causal links attributed by the user to sequences of events. Hence, a mechanism that can generate event co-occurrences on a principled basis will elicit the perception of causal

relations between these events. In return, the modification of causal laws will determine original virtual world behaviours.

This mechanism is implemented into a "causal engine", a system intercepting events in the virtual world and re-arranging their occurrence using specific knowledge about the desired behaviour of the virtual world [7]. The causal engine operates continuously through sampling cycles that are initiated by the occurrence of actions in the virtual world. Basically, the occurrence of events affecting world objects initiates a sampling cycle, during which the system recognises potential events and stores them while inhibiting their effects (it could be said that it "freezes" them). The causal engine then transforms these "frozen" events, by altering their effects, before re-activating them. This re-activation then initiates a new sampling cycle. The causal engine operates by recognising high-level events (introduced above as Context Events, or CE), whose semantic properties are used to generate new causal associations. These high-level events (such as breaking, filling, launching, etc.) are recognised from primitive events obtained from the graphics engine. For instance, the candidate CE for the glass breaking, is triggered by the glass hitting the table surface. This means that during a sampling cycle, a `break(?glass)` CE will be instantiated upon recognition of the `hit(?table, ?glass)` basic event, as the CE's conditions are satisfied. This CE will be the target for effects' modifications in the causal engine.

These modifications of CE's are carried out though the applications of specific knowledge structures, called Macro-Operators (Henceforth MOp). MOp use world knowledge (for instance on physical properties of objects) to modify appropriate CE's parameters. For instance, objects which should break up as an effect of the CE could be replaced by similar, "breakable", objects. We can illustrate the behaviour of the Causal Engine on a simple example. The test case we will consider is that of a glass being grasped, then dropped by the user for a certain height onto the surface of a table, which can also hold other objects, such as similar glasses. The default physical behaviour would consist for the glass to break on impact (Figure 4, top line), as would be directly encoded as an object behaviour in a Physics engine. Figure 4 represents several alternative behaviours. The default object of the break CE, i.e. the falling glass is substituted with the other glass standing on the table. The basis for this substitution being that the two objects are similar (actually identical), and in close spatial relation. The resulting impression is depicted on Figure 4 (bottom line, left): the glass falls on the table and upon impact on the table, it is the adjacent glass which breaks up.

From the user's perspective, the normal cause-effect sequence is disrupted: the triggering event of a given CE, in this case the glass falling on a table, will be followed, not by its default consequence (e.g. the falling glass breaking), but by an alternative effect (e.g. a nearby glass breaking without being directly hit). The causal engine can generate multiple alternative behaviours for a given CE: in this example, the table can break rather than the falling glass (Figure 4, bottom line, centre), or the adjacent glass on the table could tilt, spilling its contents (Figure 4, bottom line, right). The causal engine can generate alternative cause-effects relationships for the complete set of events occurring in a virtual world, so as to redefine the overall physical behaviour experienced by the user.

5 Conclusion and Perspectives

We have presented a new approach to the implementation of virtual world's physical behaviour. This approach is based on the simulation of physical phenomena using symbolic computation rather than numerical integration. It uses established AI techniques, such as qualitative simulation as well as a novel approaches to explicit definition of laws of causality. This approach enables the description of the overall "laws of Physics" of a given world, supporting in particular the description of alternative laws of Physics. This simulation method is compatible with the operation of native Physics engines, which can still take charge on the non-discretised aspects of the simulation (e.g. object kinematics): in that sense, even as symbolic methods, they do not compromise the response time of the overall system.

Acknowledgements

This work has been supported in part by the European Commission through the ALTERNE project, IST-38575.

References

1. Lewis, M and Jacobson, Games Engines in Scientific Research. *Communications of ACM*, Vol. 45, No. I, pp. 27-31, 2002.
2. Jiang, H., Kessler, G.D and Nonnemaker, J. (2002). DEMIS: a Dynamic Event Model for Interactive Systems. ACM Virtual Reality Software Technology 2002, Hong Kong
3. Cavazza, M., Hartley, S., Lugrin J.-L. and Le Bras, M., Qualitative Physics in Virtual Environments, ACM Intelligent User Interfaces, pp. 54-61, 2004.
4. Fikes, R. E. and Nilsson, N. J., STRIPS: a new approach to the, application of theorem proving to problem solving. Artificial Intelligence, 2 (3-4), pp. 189-208, 1971.
5. Wilkins, D. E. (1988). Causal reasoning in planning. Computational Intelligence, vol. 4, no. 4, pp. 373-380.
6. Forbus, K.D., Qualitative Process Theory, Artificial Intelligence, 24, 1-3, pp. 85-168, 1984.
7. Cavazza, M., Lugrin, J.-L., Hartley, S., Libardi, P., Barnes, M.J. and Le Bras, M., 2004. ALTERNE: Intelligent Virtual Environments for Virtual Reality Arts. Smart Graphics 2004 Symposium, Banff, Canada, Lecture Notes in Computer Science vol. 3031, Springer Verlag.