Alternative Reality: Qualitative Physics for Digital Arts

Marc Cavazza, Simon Hartley, Jean-Luc Lugrin and Mikael Le bras

School of Computing and Mathematics, University of Teesside,
Middlesbrough, TS1 3BA, United Kingdom
{m.o.cavazza, s.hartley, j-l.lugrin, m.lebras}@tees.ac.uk

Abstract

Virtual Reality Art often involves the design of artificial worlds to provide new experiences to the spectators. A recent trend in digital art has been to address physical behaviour explicitly. However, the prospect of modifying and authoring alternative physical laws demands appropriate software tools. In this paper, we introduce the use of qualitative physics in Virtual Reality Art. We discuss how qualitative process theory can be used to specify alternative physical behaviour in virtual environments. After introducing the hardware and software architecture for our project, we discuss an example which is part of early experiments with the baseline "alternative reality" software we are developing.

Introduction

In this paper, we describe an ongoing project which adapts qualitative physics to the field of Digital Arts. Virtual Reality art [Moser, 1996] [Grau, 2002] offers the perspective of creating alternative worlds that do not obey the traditional laws of physics and as such can provide novel user experience. This vision was actually part of the original ideas behind Virtual Reality, which advocated it as a way of providing new experiences, sometimes even explicitly referring to psychedelic ones [Leary, 1994].

Our current work in *Alternative Reality* investigates the definition of alternative physical behaviours that could take place in real time within an interactive environment. This would open new applications in Digital Arts as well as entertainment computing.

This is certainly an unusual application for Qualitative Physics, yet we aim at demonstrating its relevance and its ability to provide a principled solution to the design of physical behaviour in virtual environments. After a brief introduction to the state-of-the-art in Digital Arts, we describe the overall system architecture including the basic elements supporting physical behaviour. We then discuss how qualitative processes can be implemented to override normal physical behaviour and how these can be integrated into the system and give an example from work in progress. Finally, we describe early results from our work, in the form of an implemented example.

Digital Art works are often based on sophisticated briefs taking advantage of the plasticity of the digital medium to free themselves from the traditional constraints of the physical world. But more interestingly, in recent years, the work of several artists has explicitly addressed Physics as a source of inspiration, in particular where it could depart from our everyday physical experience. For instance, the exhibition "The Amplitude of Chance" in Kawasaki was entirely devoted to briefs exploring causality [Sato And Makiura, 2001].

One sophisticated and even more explicit example is constituted by the animation series "The Quarxs™", which features a set of invisible creatures which violate the laws of physics and provide an explanation for the unaccounted phenomena of our everyday world. These "insect-like" creatures are named after the physical laws they twist: for instance the *Reverso Chronocycli* causes time to flow backwards and the *Spiro Thermophage* (Figure 1) lives in water pipes causing cold water to always precede the flow of hot water.

Figure 1 . The Spiro Thermophage from the Quarxs™ (© Maurice Benayoun)

As the example from the Quarxs™ illustrated, the modification of physical laws is a principled way of creating alternative universes. In the case of alternative reality, these modification should be effective within a real-time virtual environment, rather than an offline

animation (hence the term "reality"). The definition of new world behaviours often starts from an artistic brief. Not only is this qualitative in nature, but the changes tends to be initially described in terms of the abnormal behaviour, rather than in terms of the principles that would underlie these changes. It would be rather challenging to devise equations reflecting these changes and unclear how structural equations that could propagate causality would be defined in this context.

For all these reasons, we have turned to Qualitative Process Theory [Forbus 1984] as a framework for qualitative physics modelling. The fact that it is centred around processes offers the possibility to define and to integrate behaviours at different levels of abstraction, integrating them with the interactive aspects of virtual environments.

System Overview

Virtual Reality Art is most often presented in the form of installations using immersive displays such as CAVEs™. This supports the participation of a limited number of spectators enjoying some mobility (over a constrained space). Our target systems are large-scale virtual reality installations, based on the SAS Cube™, which is a 4-wall, PC-based, CAVE™-like, immersive visualisation system. We use a game engine, Unreal Tournament 2003™ as a visualisation engine and as a development environment. Game engines are now increasingly used for visualisation in scientific research due to their rendering performance and their ability to communicate with external software modules [Lewis and Jackobson, 2002]. In addition, Unreal Tournament has previously been ported to CAVE™ systems [Jackobson and Hwang 2002] and we are currently porting UT 2003™ to the SAS Cube, using the original approach described in the CAVE-UT implementation.

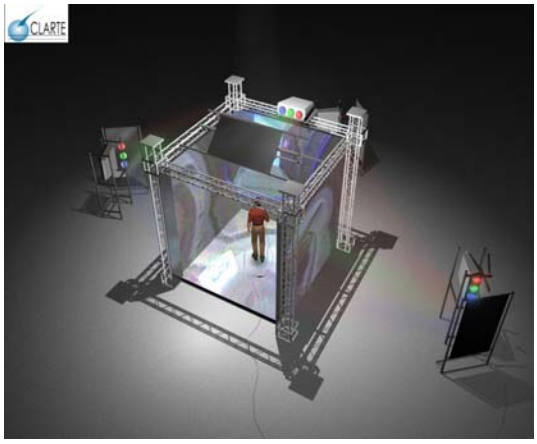

Figure 3. The SAS Cube™

The rationale for using a game engine is not just its graphic rendering abilities or the built-in mechanisms for user interaction with world objects. The Unreal Tournament 2003 engine incorporates a physics engine (the Karma™ system from Mathengine™) together with a sophisticated API and programming features supporting the modification of baseline physics. Traditionally in game engines, physical behaviour of objects can be directly modified using mutators, which are procedures (in the games' Java-like scripting language, Unrealscript™) behaving as "mini-mods" (where a mod is a modification of the gameplay, i.e. a consistent alteration of standard game properties). However, overriding traditional physics with Qualitative Physics processes requires more complex developments, though their implementation can still make use of mutators as a low-level mechanism.

The software architecture integrates an external Qualitative Physics module, developed as a standalone C++ software, which simulates physical behaviour using a range of qualitative physics processes, with the Unreal Tournament 2003 engine, which supports 3D visualisation and interaction. These modules communicate via a TCP/IP socket interface by exchanging values for qualitative variables corresponding to properties of the virtual world's objects. However, the actual integration of qualitative simulation in the game engine in terms of objects' transformations relies on low-level features of the game engine, such as the events management system. One of the reasons is that physical behaviour depends on interaction between objects and/or physical interaction by the user.

The system uses an initialisation module, implemented using Unreal's mutators, to generate the runtime application interface between the Qualitative Physics engine and the virtual environment. This initialisation module identifies virtual world objects that are associated with qualitative variables corresponding to the Qualitative Processes defined for a given application and parameterises event generation mechanisms accordingly, so that dynamic interaction can activate the relevant QPs. For instance, context events corresponding to the pre-conditions of a QP can be used to trigger its simulation.

The initialisation module dynamically generates, sets and activates the qualitative process controller, which will act as an interface (on the game engine side) between the world objects and the QPs that affect them. Once instantiated, it will start to supervise the interaction between the virtual world objects contained in the environment and the QPs running within the external Qualitative Physics engine. In addition, it can be noted that the use of a discretised simulation method is fully compatible with the basic visualisation and interaction mechanisms of a game engine, which at their lowest levels

tend to follow a similar logic for reasons of efficiency (event modelling, keyframed animations).

The Event-based Architecture

Most interactive systems are based on the notion of event for their implementation. Highly interactive systems, such as VR systems [Jiang et al] and game engines in particular intensively exploit this notion for their implementation. On the other hand the notion of event is also the basis for the high-level description of physical behaviour, as events discretise the continuous motion of objects (in terms of positions, trajectories, contacts with other objects) into meaningful high-level actions.

This section describes the event management mechanisms, which support the redefinition of alternative behaviours for the virtual environment. After introducing the native mechanisms provided by the UT game engine, we will show how these can be used to define complex events corresponding to object behaviour as well as a control strategy to override the basic mechanisms supporting the world physical behaviour. This will then open the possibility of extensive redefinitions of the virtual world behaviour, supporting alternative views on physics and causality.

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 for the environment. Formally, an event can be characterised as an encapsulated message, which is generated by an *Event Source* and sent to one or more *Event Consumers*, these being both objects of the environment. The transmission of an event to an Event Consumer triggers some specific action in response to that event. It should be noted that the actions triggered by Event Consumers can further be detected by other Event Sources, and this accounts for the possibility of Event propagation as a kind of "forward chaining", propagating the consequences of an action.

The Unreal Tournament Engine implements two different kinds of event sources: the primitive events, which are low-level events defined within the game engine and the programmed events. The latter are events whose definitions are scripted (i.e. can be programmed by the system developer) and are used to customise the interactions between objects, by defining which objects will trigger (or react to) specific events. The basic events can be classified into six major categories, of which two are mostly used in our implementation to redefine environment behaviour: the interaction events and the time notification events. The interaction event category is further refined into several sub-categories, the most important being: *physics and world interaction event*, *player input event*, and *trigger event*, which is the basic event class supporting the definition of high-level event.

On the other hand, time notifications are related to the engine internal time management system and can be used to define the control cycle of any new event-based layer, for instance to program the sampling rate of event management.

From another perspective, basic events can also be classified as discrete or continuous events. Discrete events notify instantaneous actions, such as "bumping", while continuous events signal the beginning and ending of durative actions, for instance touch/untouch, attach/unattach (used for instance for carrying or manipulating physical objects).

The redefinition of event mechanism comprises three main aspects: i) the attachment of events to objects, ii) the overriding of native event generation mechanisms and iii) the definition of ad hoc complex events from basic ones.

Relating events to objects is implemented using the native UT *mutator* system, which in UT supports the redefinition of object behaviours. This confers a new behaviour to the environment objects, which is the ability to enter into an "event interception" state. Once an object is set into the event interception state, every event fired for this object will not trigger any immediate action (via the procedure described above) *i.e. the message coming from the event source will be intercepted at the event consumer level*. Instead it will use a procedure to signal the event call and arguments to an Event Controller module, such as those we have defined to implement qualitative physics (see below).

The Unreal Engine relies on a fixed set of basic events. However, for most applications it is necessary to define high-level events, whose semantics is dictated by the application. Such complex events are often called *context events*. Because they are aggregates of basic events, context events can be recognised by parsing a stream of lower-level events using a template for the (high-level) event to be recognised. This is a standard approach in event recognition, which has been used previously in computer vision and VR alike [Andre et al 1988] [Cavazza and Palmer 1999]

Software Architecture for Qualitative Physics

The virtual world objects that are manipulated by Qualitative Processes are, in fact, derived from a special class of Unreal Objects (the class normally used to represent physical objects) called Qualitative Physics

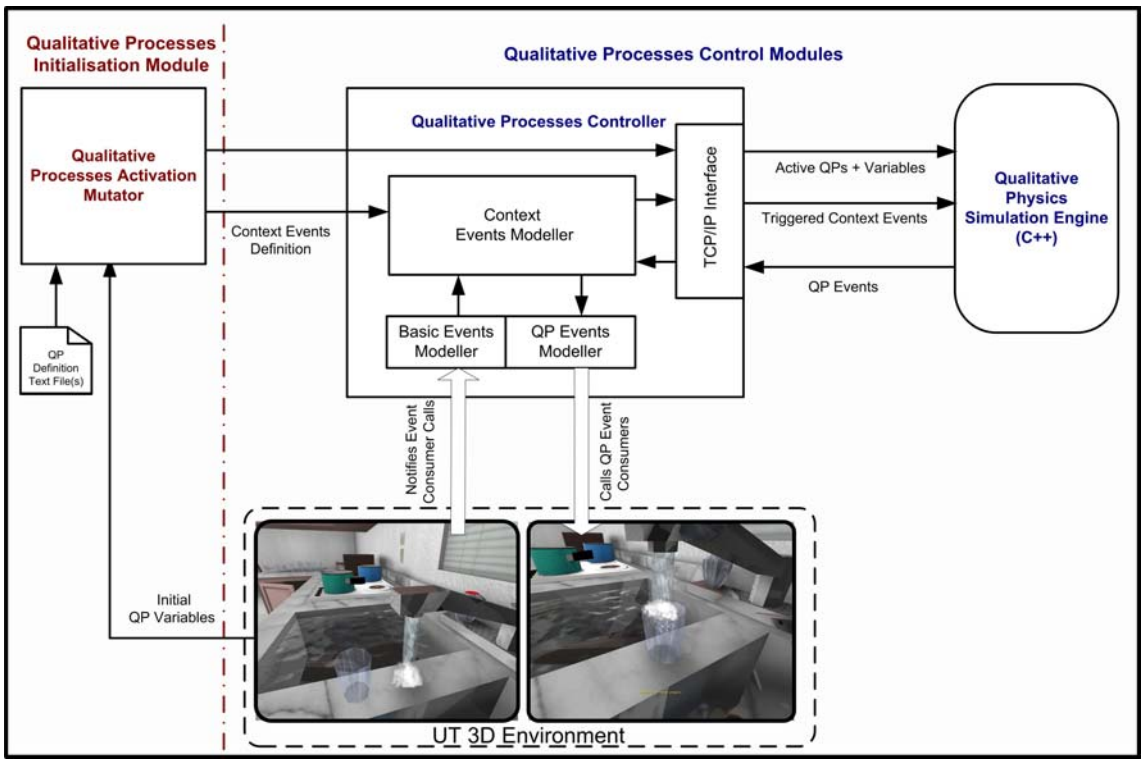

Figure 2: Qualitative Process Control Software Architecture

Object (or QP objects). This object class has three main characteristics. It owns "QP Variables" (variables that can be altered by qualitative processes, which characterise dynamic physical properties of the object), and is associated an event interception system (to activate relevant qualitative processes) as well as a set of "QP Events", which are events transforming the object according to the output of qualitative processes.

The Event interception system is constructed over the Unreal native event system as described in the previous section. In particular, the Event consumers (Native Event Function) of the QP object have been redefined. The events recognised correspond for instance to the pre-conditions of a qualitative process (e.g., that a recipient be aligned with a flow of liquid). Every event fired for this object will be communicated to the Qualitative Process Controller.

QP Objects define special events triggered by the Qualitative Processes, which are called QP Events. The QP Events' implementation and signature depends on the type, nature and state of the QP objects they're associated to. For instance, certain QP events will launch specific animations (e.g. water level rising, object melting or breaking), while other QP events will potentially update an object's variables (e.g. mass) or its state.

The Qualitative Process Controller is composed of four components: the Basic Events Modeller, the Context Events Modeller, the TCP/IP Interface and the Qualitative Process Events Modeller.

The Basic Event Modeller continuously intercepts and stores the basic events triggered from the QP object instance interacting in the virtual environment. It passes the list of events to the Context Events Modeller at regular intervals defined by an "event sampling rate", which depends on the kind of processes defined for any given application.

The Context Event Modeller contains the definition of the Qualitative Process Context Events, i.e. high-level events that are part of qualitative processes definitions, such as events corresponding to the pre-condition of a qualitative process. These are defined at initialisation time as Finite-State Automata whose elementary nodes are basic events,

so they can be dynamically recognised by parsing a sequence of basic events involving relevant objects.
The triggered context events activate the simulation of Qualitative Processes, which takes place in the external Qualitative Physics engine. Throughout the simulation the active qualitative processes update the qualitative process' variables, duplicated from the virtual world during the initialisation phase. When a process reaches Landmark values, it forwards appropriate values to the Qualitative Process Controller via the TCP/IP interface.
Within the Qualitative Process Controller, the QP Event Modeller converts the information coming from the Qualitative Physics engine into appropriate events that would be dispatched to the corresponding virtual world objects. These events will update the objects' appearance according to the variables' new values: for instance a heating object would glow, the level of water in a recipient would rise, etc.

Formalising Qualitative Processes for Alternative Physics

In this example we detail how we are using the definition of a basic heat flow process as in [Collins and Forbus, 1989] and altering some of the basic definitions of a Heat-Path. Where A Heat-Path is described in terms of objects in contact and heat-aligned indicates that the contact is unbroken as in [Collins and Forbus, 1989] as Shown in figure 1.

Process: Heat-Flow (?src ?dst ?path)

Individuals: ?path :type heat-path
:conditions (thermally-connects-to ?path ?src ?dst)
?src :A simple-thermal-physob
?dst :A simple-thermal-physob

Preconditions: heat-aligned ?path

Quantity Conditions: (A(temperature ?src)>
(A(temperature ?dst))
Relations:
(Quantity heat-flow-rate) α
(A(temp ?src)- (A(temp ?dst))

Influences: I+ (heat ?dst) (A heat-flow-rate)
I- (heat ?src) (Aheat-flow-rate)

Figure 3: QPT Formalisation of a Flow Process

The heat path that is described uses the Thermally-Connects-To to describe a one way thermal connection between the basic physic objects, Heat-Connection indicates a bidirectional thermal connection and the Thermal-Conductance is the paths ability to transmit heat. In our initial implementation we considered the heat path to be connected when the two objects for which they relate a heat path receive the context event. This aligns the heat path in the same way as the fluid path.

In an extension to this method we are altering the heat-aligned path precondition to determine which heat flow process occurs. In the initial implementation the context event aligned is generated when the heat physics objects bump into one another.
In an alternate method the context event is generated for an aligned connection of the heat path for the Objects, not between the two objects that are in connection, but between the heat source and a heat destination that is distant. This method allows the heat process to act at a distance.
Another alternative implementation for the generation of the context event allows the creation of the context event between two heat objects whose heat potential is the greatest. This allows the different heat process to occur at distance. Since we have not altered the heat flow or boiling processes the system can heat and boil the heat destination.

Figure 4: Heat Flow with alternate Heat Path

Results

We created a test environment to experiment with this qualitative simulation technology. This environment was inspired from some of the Quarxs™ episodes, which tend to take place in everyday settings such as kitchens and bathrooms. Actually these areas concentrate many potential physical processes, such as the fall of objects, flows of

liquids, heat transmission, surface contacts, formation of bubbles and foam, etc.

To illustrate the definition of alternative qualitative processes we will discuss an example in an area well described in qualitative physics, including by qualitative process theory, which is fluid behaviour. Let us imagine that we want to create recipients in which more liquid can be poured than its volume would normally allow. We want this to take place without violating other physics laws such as conservation of mass. We will thus define a process according to which when more liquid is added, the quantity of matter will increase but not the volume, which amounts to make density of that specific fluid to vary locally, in that recipient, according to the volume poured into it.

Below is an example of the fluid flow process that we implement to show the alternative behaviour of a glass, which can hold an infinite amount. This alternative behaviour leads to the glass becoming to heavy for the user to move.

The description of the fluid flow process which describes the filling of the Glass :

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 5: QPT Formalisation of a Fluid Flow Process

In our example we have defined three Objects for the qualitative physics simulation. The details for these objects are then sent to the Qualitative Physics engine which instances the Processes fluid flow. The fluid process is instanced as the three objects which its applies to exist i.e. the Tap (a contained liquid source), the Glass (a contained liquid destination.) and a fluid path (shown as a column of water). This then generates for the list of potential processes two fluid flow processes, these processes represent water flowing processes from the tap to the glass and from the glass to the tap.

Next the simulation loop for the qualitative physics engine is given for the example of the water flowing into the glass. The definition of a process determines the conditions under which it can become active. The conditions that the user can directly affect are known as preconditions for instance aligning a glass/container under a tap is a precondition for water flow to occur. (as shown in Figure 7. Process Generation and user interaction). Context Event is defined by adding the QP FlowPath Mutator to the Event controllers list of Active Context Events, where A, B and C are virtual world object references. Note: The object A points to a FlowPath Object instance in the world (i.e.: Tap Water flow) and B refers to a Container object instance (i.e., a Glass). It is the alignment precondition that the glass has to be aligned with the flow that the moving the glass by the user fulfils. This movement causes the context event path aligned to be sent to the QP engine.

The conditions that apply to the quantities within the objects, whose values and changes are governed by the qualitative process theory, are known as quantity conditions. An example of this is for the water flow process for which the pressure of the individuals needs to be different for the process to occur. To determine which processes are active we have to test the preconditions and the quantity conditions if these are both active we place these objects into the active process stack. In our example, we had two potential processes which both had passed the quantity conditions. Now for the process to occur the precondition and quantity conditions for the processes need to be tested. For both processes the precondition are that there must be an aligned path between the two contained liquid individuals. If the user then moves the glass under the flow we have the context event aligned path that is sent from the environment to the QP engine. This event aligns the flow and allows both processes to pass the preconditions. The next stage is the quantity conditions for the processes since we are comparing the pressures for the two objects only one process will pass this stage, the process from tap to glass.

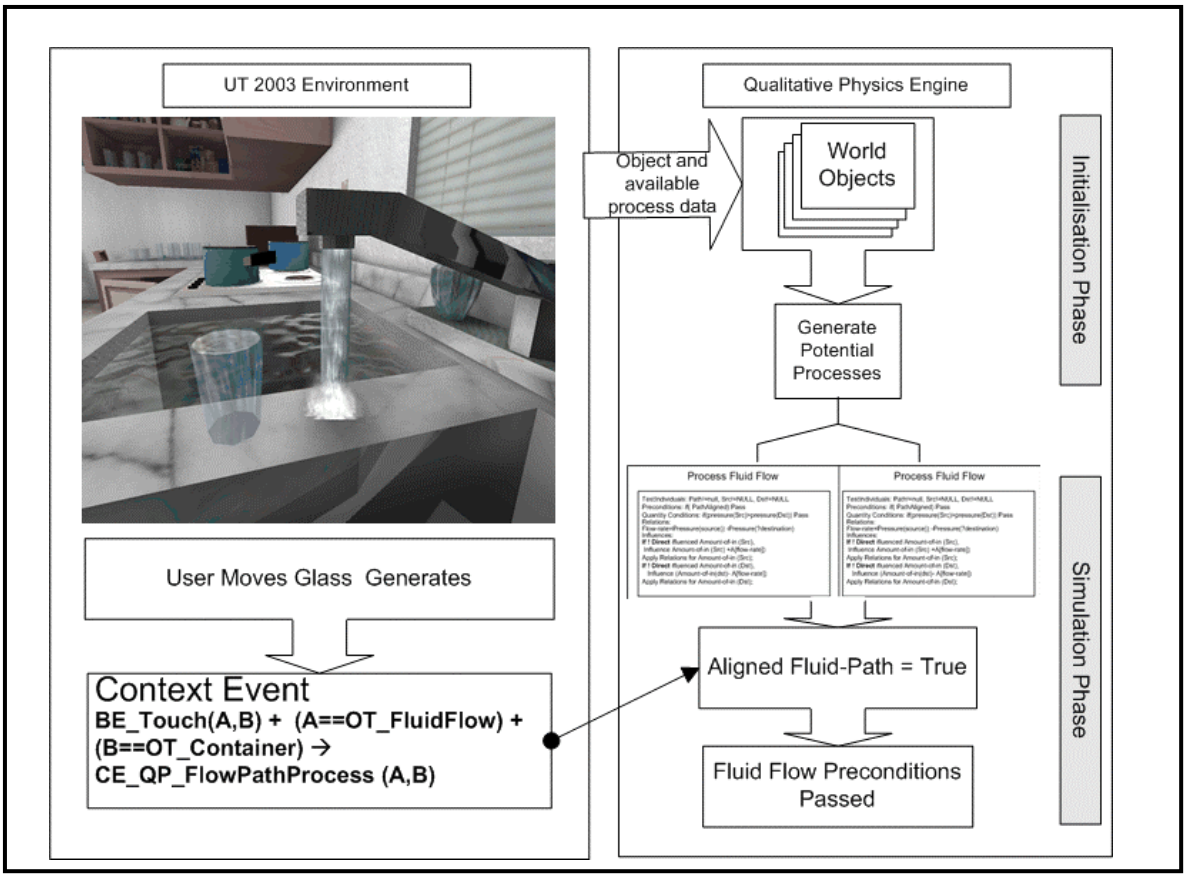

Figure 7: Process Generation and user interaction

The process will then become active (as shown in figure 7: Process Activation) and begin altering the quantities as described by the influence equations.

In the case of water flow the amount of water in the glass will increase.

The next stage involves calculating the relations between the individuals for the process and then determining changes in the quantity via resolving influence equations.

When determining changes in the quantity via resolving influence equations the first stage is to resolve the direct influences of the processes. In our example the directly influenced values are the amount of water in the glass and the amount of water in tap. Then the process applies the relations for the changes in these quantities, this is to alter the indirectly influenced quantities. In our case we are changing the amount of water in the glass but by our alternate laws the glass can hold an infinite amount. So the water will not over flow but the glass will get heavier so the user will be unable to move the glass. Eventually a limit point for the glass/container mass will be reached and this will generate an event that will be sent from the qualitative physics engine to the UT 3D virtual environment.

This event informs the virtual environment that the mass has passed a certain value. For alternative laws the virtual environment could respond in many ways to this event such as breaking the glass or starting different processes any of which would affect the user experience.

Conclusions

Modifying the laws of physics in Virtual Reality to create principled, yet not always consistent, behaviours is certainly a challenge in terms of conception, implementation and authoring. Qualitative physics, in particular qualitative process theory can support the definition and the authoring of a wide range of alternative behaviours. Qualitative physics also facilitates integration within an interactive environment, being compatible with

the timescales of user interaction and the animations that visualise physical events.

Acknowledgements

This work has been funded in part through the ALTERNE project (IST-38575), funded by the European Union under the Information Society Technologies programme (Cross-Programme Action 15). Maurice Benayoun is thanked for introducing us to the Quarxs™ and for authorising the use of Figure 1. SAS-Cube™ picture courtesy of CLARTE/Laval Mayenne Technopole. Unreal Tournament™ is a trademark of Epic™ Ltd.

References

Andre, E, Herzog, G and Rist, T, 1988: On the Simultaneous Interpretation of Real-world images and Natural Language Descriptions: the SOCCER system. Proceedings of the 8th
European Conference on Artificial Intelligence, Munich,

Cavazza, M. and Palmer, I.J., 1999: High-level Interpretation in Virtual Environments, Applied Artificial Intelligence.

Collins, J. and K. Forbus. (1989). Building qualitative models of thermodynamic processes. Proceedings of the Qualitative Reasoning Workshop.

Forbus, K.D., 1996: Qualitative Reasoning. In A.B. Tucker, editor, The Computer Science and Engineering Handbook, pages 715--733. CRC Press.

Grau, O.,Virtual Art, 2002 : From Illusion to Immersion ,Cambridge (Massachussets), MIT press. ISBN: 0262072416.

Jacobson, J. and Hwang, Z. 2002 Unreal Tournament for Immersive Interactive Theater.
Communications of the ACM, Vol. 45, 1, pp. 39-42.
Jiang, H., Kessler, G.D and Nonnemaker, J., DEMIS, 2002: a Dynamic Event Model for Interactive Systems. ACM Virtual Reality Software Technology 2002, Hong Kong

Lewis, M and Jacobson, Games Engines in Scientific Research. Communications of ACM, Vol. 45, No. I, January 2002. pp27-31.

M. Sato and N. Makiura,2001 *Amplitude of Chance: the Horizon of Occurrences*, Kinyosya Printing Co., Kawasaki, Japan.

Leary, T,1994: Chaos and Cyberculture, Ronin Press.

Moser, M.A. (Ed.), .,1996, Immersed in Technology: Art and Virtual Environments, Cambridge (Massachussets), MIT Press.