*ICAT 2003*
*December 3-5, Tokyo, JAPAN*

# A Formalism and a Tool for Diverging Requirements in VR Scenario Modeling

Richard Wages, Benno Grützmacher, Georg Trogemann

Laboratory for Mixed Realities
Am Coloneum 1, D-50829 Cologne, Germany
*{wages, gruetzmacher, trogemann}@lmr.khm.de*

**Abstract**

Any VR scenario should offer a certain degree of non-linearity or openness regarding the story progress as a consequence of the interaction with the user. In this paper we propose a new formalism, which we regard as suitable for authors and artists to develop compelling and flexible interactive stories. We distinguish two levels of openness in storytelling. The first level of openness deals with the freedom of the author. Storytelling is regarded as an ill-defined problem and hence Authoring Tools have to support the author defining his goal and to iteratively describe, develop and improve his idea of the story. The second level of openness deals with the freedom of the user. From the perspective of the audience of an interactive story we want the plot to behave in personalized ways, i.e. depending on the actions of the user the story should develop in different directions. In accordance with these two types of openness we developed a prototypical graph based tool for the creation of non-linear scripts to make VR scenario authoring accessible for a much greater number of creative people.

**Key words**: Virtual Reality, Non-Linear Scenario, Story Modeling Formalism, Parallel Graphs, Authoring Tool

## 1. Introduction

Obviously VR scenarios cannot be strictly composed the same way as for example linear movies with entirely pre-produced images. Authors of non-linear scenarios have to describe their intentions for the progress in form of a set of explicit rules which govern the (long term) behavior of the system at runtime. Hence from an authoring point of view the subsequent challenges are twofold:

In the first place one has to provide authors with a powerful formalism which is unambiguous and can be translated directly into the VR engine of the target platform. Secondly one has to observe that VR scenarios are still predominantly designed by programmers since most of the potential authors of VR scenarios or interactive stories are not familiar with programming languages. Hence these authors also have to be equipped with an authoring tool to create such non-linear scripts for interactive stories. To meet this challenge we looked upon the once development of abstract formalisms in software engineering like statecharts (Harel charts) which among other had a strong influence on the specification of UML.

In contrast to past developing processes of 'productive' software which began with a definition of strict and *well-defined* requirements for the application and the design for an optimized implementation nowadays approaches for software development are of a more evolutionary character. This general shift is true in a double sense for authors of scenarios which are expected to show a greater openness at runtime. An author will most likely add additional objects, story fragments or entire subplots at any time during the elaboration of the scenario. This might especially be the case towards the end of the creative process when she or he gets a better feeling for the effects of the developing story. The discovery of surprising scenario behavior might even delight its creator if it contributes to dramaturgy and suspense in a way which was not necessarily anticipated. Hence we consider *testing opportunities* during any time of the VR scenario authoring process as crucial.

For our tool we chose a working sheet with the opportunity to build up parallel hierarchical directed graphs since both programmers and artists agree on such a basic authoring method, even though for different reasons: While the latter see an intuitive depiction of a proceeding non-linear story, software developers rather perceive a direct visual representation of objects which can easily be translated into finite state machines (FSM). As an example: On entering a node which nests a hierarchical lower (sub)graph programming paradigms call for an instant descent to a child node of lowest hierarchical rank. However such a procedure is an unnatural approach for an author who more likely will situate certain events to happen within this node before an optional entering of the nested (sub)story.

Our developed VR authoring tool addresses a mediation between these diverse realms. On the one hand authors are enabled to creatively build up their envisioned complex VR scenario progress to – what we consider to

be – a maximal possible extent of freedom. On the other hand they are bounded insofar as they cannot evade the use of the inbuilt (particularly easy) scripting dialogues which will lead to a logically consistent and working scenario. Thus they are also obliged to explicitly write down the results of particular events or user interactions. This means an additional advantage within the communication process during a production compared to mere textual outlines. Programmers now receive a complete scenario script in form of an XML file as an output of the tool. An interpreter for a target platform only has to be developed once. Beside the storyboarding functionality the most important component of the authoring tool is its *internal story player* for scenario testing opportunities. This player is completely textual and hence independent of the existence of any modeled images or sounds.

## 2. Related Work

Within the realm of computer games FSMs are widely used to model character real-time decision and action behavior graphically. Tools like for example DirectIA [2] and SimBionic [9] provide a framework to develop systems in a very state-oriented way. Most of the resulting control flow is governed by states or conditions of objects or processes. Yet on the bottom line both are rather implementation tools to help programmers to create popular constructs than design tools. In contrast our tool focuses on a high level scenario design itself.

An impressing example for interactive storytelling in virtual worlds is the *Façade* project [7]. Here the behavior of the *believable* characters is authored by a special scripting language (ABL). While this system attends to create a dramatic story progress at real-time again the very authoring of the character behavior has to be done by a person with programming skills.

Two prominent tools that are designed especially for the development of story are Dramatica [8], a tool which was used to develop the scripts of numerous movies, and Storyspace [3], a tool to create non-linear hypertext novels. Although the creation of scenarios is actively supported by these tools they are not applicable for VR scenario modeling since they deliver output in unsuitable formats, i.e. pure text and hypertext respectively. For our tool we decided to use XML as an extremely general output format which potentially can serve as in input for a great range of VR and game engines.

## 3. Interactive Storytelling as Two-Level-Openness

While most of the problems that where solved with computers in the beginning of computerization belonged to the field of well-defined problems, today's applications are more and more penetrating the realm of ill-defined problems. Well-defined problems are characterized by situations in which the starting position, the goal and the set of possible actions are known in advance. This means especially, that with each problem we are given some systematic way to decide, when a proposed solution is acceptable. Well-defined problems are static in some way, i.e. with the precise description of the problem at the beginning of work the whole space of possible solutions is already completely determined. Typically mathematical problems are well-defined problems.

Ill-defined problems on the other hand are characterized by moving targets. For example, all design problems and art works belong to the universe of ill-defined problems [4, 5]. The starting situation of designers and authors in creating artistic work maybe summarized best by Steven D. Katz: *'... how does the artist use visualization? ... The answer is that the artist rarely has a specific goal in mind when he begins to work, so that the process of visualization is actually the search for a goal rather the attainment of one.'* When dealing with ill-defined problems much of the work has to be invested in finding and describing the problem. With Poincaré: 'The question is not, *what is the answer?* The question is, *what is the question?'* One other feature of ill-defined problems is that the borderline between the relevance or irrelevance of information is very vague. The context of the application emerges throughout the working process and the determination of whether some information or action should be considered as relevant or irrelevant can only be decided during working on the problem and not beforehand. Ill-defined problems are open in the sense, that there are no definite criteria to judge the solutions. This means, that ill-defined problems are not really *solved*, rather the process of working on the problem is stopped at some point, were the state of the solution is 'satisfying'.

From what we have said so far, it should be clear, that storytelling and the creation of a rich interactive scenario belong to the realm of ill-defined problems. The art of storytelling has no standardized solutions, sometimes no requirements exist at all, the goal becomes only clearer *during* the working process, and the process itself is extremely iterative. Also the criteria for success in the artistic context of storytelling cannot be formally described but works have to be evaluated by unsharp aesthetic means. Hence one important requirement any authoring tool for storytelling has to fulfill is to support the creative solution of an ill-defined problem.

But regarding storytelling as an ill-defined problem is only the first level of openness. In the field of storytelling and interactive scenarios in general, we have to deal with even a second level of openness. While the first level deals with the freedom of the author during the design process of a story, the second level deals with the freedom of the user. From the perspective of the audience of an interactive story we are also interested in some sort of openness. We want the story to behave in personalized ways, i.e. depending on the actions of the user the story should develop in different directions. This

second level of openness also has to be supported by the authoring tool.

The implemented formalisms, features and functionalities of our authoring tool aids 1st. and 2nd. level openness of scenarios in multiple ways. Since the design of the tool functionalities and the underlying formalism for scenario authoring are intertwined we describe them not one after the other but in accordance to the respective level of openness they support during story authoring.

## 4. Support of first level openness

To support first level openness the Authoring Tool must address artists rather than programmers. The tool should therefore be easy to use and not demand any technical skills or programming languages. Again it is the domain of computer games which gives a hint in the growing problem in the endeavor to create environments of rich experiences for the user. Computer game developers search for ways to include non-programmers on the

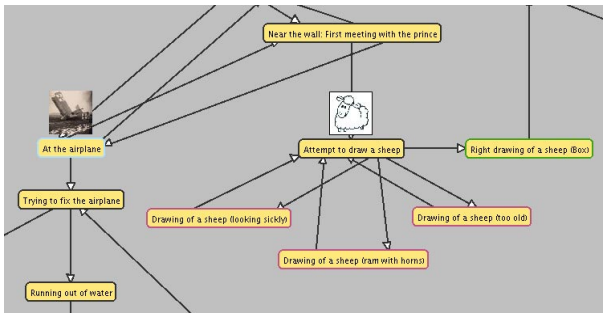


Fig. 1 A Fraction of the Storygraph

development team (level designers, authors) in the creational process and use the easy to visualize FSM diagrams for the communication of system progress over time. Since we will finally need an even more general structure we decided to offer a working sheet with the opportunity to build up *parallel hierarchical directed graphs* as basic elements.

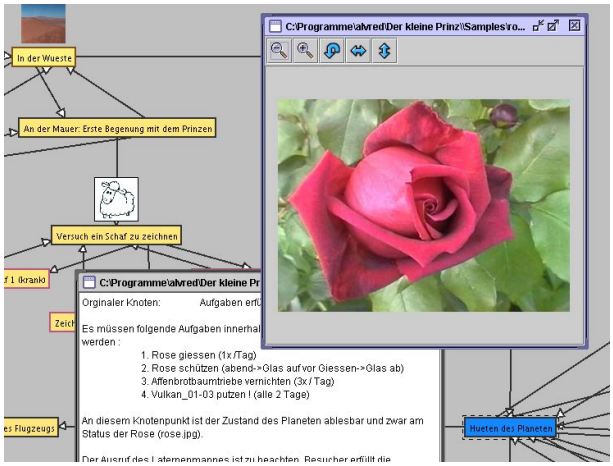


Fig. 2 Internal Media Player

Not only can authors visualize the progress of the non-linear scenario (*storygraph*) but also use the tool as non-linear storyboard for themselves as well as to provide information for the further production process. Moreover the storyboard functionality is enhanced by an internal media player to illustrate sample files of various file formats.

To cope with growing size and complexity and to overall organize huge structures one can recursively encapsulate substructures within nodes. A better transparency is given by the possibility to access to all hierarchical levels of story description directly at any time.

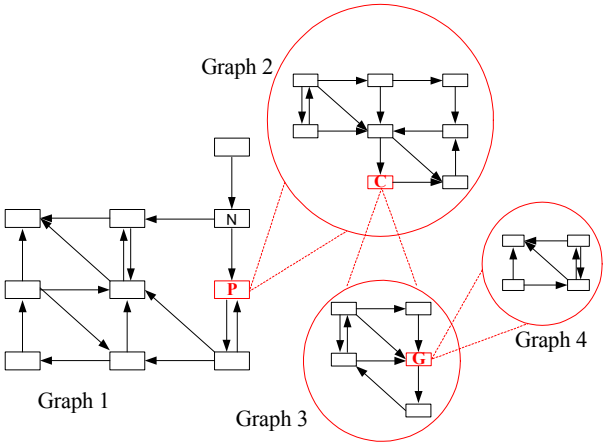


Fig. 3 Hierarchical Graph

With the described features authors are enabled to build any desired scenario progress from scratch if they like to. In addition the tool can be equipped with typical structures and special objects (*building blocks*) of a given target platform. These structural templates were not only introduced for economical reasons but also to aid the inexperienced author and give her or him something to start off right away.

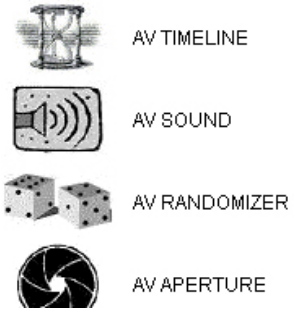


Fig. 4 Building Blocks

As a direct feedback is essential especially at early stages for any artistic and creative work to see if the realization of ideas is working out as expected. The tool allows the author to process the story in an *internal story player*

before any images or models are produced.

At the end of the successive process of forming the story from a vague idea towards the final product, the tool ensures that the scenario is broken down into a stringent form. The generated XML file is a formal description of the scenario. Due to the generality and popularity of the XML standard the output may potentially be used by a wide range of target platforms.

## 5. Support of second level openness

In the following we propose the approach of bringing more openness to the interactive scenario by introducing *parallel* graphs. By this we take a further step to extend the formalism of pure FSMs. Similar extensions of FSMs have been made earlier among other with the introduction of concurrency as for example in Harel charts [6]. As pointed out in section 3 the tool as well as the underlying formalism also address the demand to allow for this creation of a more open scenarios.

Given a system environment which is capable of generating a virtual world along scripted rules like a VR or game engine. Obeying the aimed aesthetics or logic of the presented scenario, some or all control has to be transferred to more or less autonomous system entities in a meaningful way. Hence there is a need for a central system entity which somewhat strongly enforces the author's intentions on the progress of the running scenario comparable to a conductor or moderator. At runtime all these entities together with the user govern the progress of a particular scenario.

It will always depend on the aimed openness of the system to which extent the author is willing to transfer or even give up control. An authoring approach, which is based on the idea to distribute the control among intertwining system entities – the central and most powerful of which is the representation of the author's intention regarding the progress itself – provides for a great spectrum of conceivable scenario types.

In our case this central entity will be the above

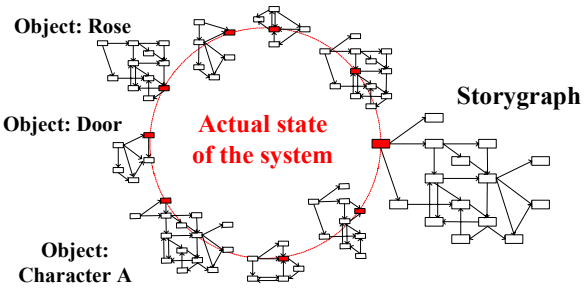


Fig. 5 Parallel Scenario Graphs

mentioned storygraph. All other entities (*object graphs*) will be of the exact same structure and can be authored the same way, but the storygraph will play the prominent role. When we offer to the author the possibility to create object states we primarily think of *story relevant* states rather than for example physical states. In a running scenario changes of the system correspond to a transition in at least one of the graphs. Such a transition within a graph can be triggered by a clearly defined condition like a particular reached state of a different graph or a certain state constellation of a collection of graphs. The actual state of the system is then determined by the collection of all active nodes of all objects.

Table 1 Node Content

| Node Content | Function |
|---|---|
| Storyboard | Textual description of a scene / state represented by this node |
| | Media samples to support the description (e.g. pictures) |
| Story Logic | Objects which are referenced within this node |
| | Parameters characterizing this node |
| | Event Scripts belonging to this node |

To expand the authoring possibilities in addition to discrete states each node can be equipped with (continuous) *parameters*. After serving as a more or less simple container for the author's description of the envisioned scenario, the nodes will be used as the place to author the logical interdependencies between the different graphs. This is done by the author in form of event scripts. To create such an event script the node has to reference other objects. An event script is then a simple logical if-then expression, where events can be defined as a condition to trigger other chosen events as results.

Conditions (c1, c2, c3,..) → Results (r1, r2,..)

Event scripts can only be built with the help of a dialogue which only allows for conditions and results (i.e. transitions) which are actually possible within the scenario since they were earlier created by the authors themselves. Thus authors are obliged to break down the scenario into a concrete set of rules for the system. Although the tool cannot completely guarantee the consistency of a created story it thus supports the authoring of a reliable scenario to a great extent.

For parallel processes the priorities of execution and the communication between the parallel entities are crucial. As pointed out above any state of an object can be scripted as a condition of an event script. The condition is considered as true in the moment when the appropriate state is taken leading to a processing of all the results of this script. Since any transition in any graph could potentially trigger a script, every transition can be

considered as an event. To manage this amount of events, a mechanism applies which can be described as follows: Given three instances of a list

*Event Scripts List (ESL):* Contains all event scripts of the nodes which are active at a time and therefore listening to occurring events.

*Object states list (OSL):* Contains all objects and their current states.

*Processing List (PL):* Contains all event scripts which will be executed because they were matched. An event script is matched when all its conditions are true at the same time evaluated by comparing ESL and OSL. While processing the PL all results of an event script will be processed after another, before the processing of the next event script will be started. The list will be processed from top to bottom (the oldest entry on the list is the first to be executed and will be deleted afterwards). During processing the list new entries can be registered as consequence of another result.

Starting from the execution of a result (e.g. processing the result $R_{j,i}$ of an event script $S_j$ leading to a transition to node N in a graph G) on the PL will lead to the following process execution order:

1. Transition to node N
2. Update OSL & ESL
3. Write start states of objects of N in PL (as event scripts)
4. Write start scripts of node N in PL (event scripts where the condition is true in the moment the node N in entered)
5. Evaluate matching scripts (by comparing conditions of the scripts on ESL with OSL). Put matched scripts on the PL.
6. Process next result of Script $S_i$ (return to 1.)
7. Delete the event script $S_j$ on the PL. Process first result of next event script $S_{i+1}$ on the PL. (return to 1.)
8. Idle (Interaction of the user or time triggered events are new transitions to return to 1.)

Nonlinear time management:

For the dramaturgy of storytelling the management of the temporal occurrence of events is a fundamental element which ought to be under the control of an author. Even if the user has only limited control over the progress of the scenario, his freedom leads normally to the destruction of a meaningful time management. To allow the author somewhat more influence on the temporal progressing of the story *timelines* as special objects are provided in the tool. These timelines can be referred to in any node and integrated into the event scripts' logic to trigger transitions in the storygraph or in objects. Especially the following characteristics of timelines are considered as crucial for nonlinear time management:

*Flexibility:* Because of the full integration of timelines in event scripts a timeline can be started, paused and stopped under any possible condition (e.g. any state of any object). Likewise a timeline can also be scripted as a condition itself, leading to time triggered transitions.

*Hierarchy:* Another important feature of timelines lies in the combination with the concept of hierarchy and the possibility to define an event script as local or global. By defining a time triggering script as local or as global the author can also define time management on different hierarchical levels.

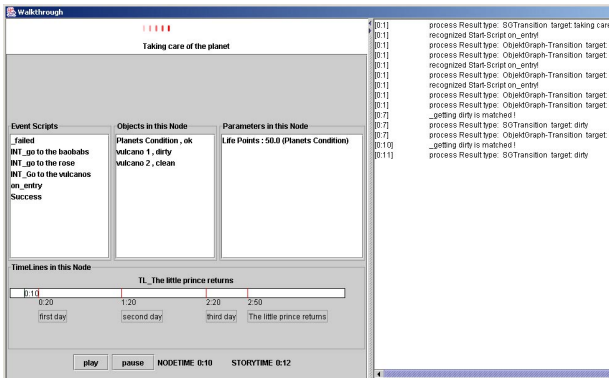


Fig. 6 Internal Story Player

One has to keep in mind that even simple rules are capable of generating complex behavior [10] that cannot be anticipated completely by any author on a descriptive level. Hence in practice extensive testing, i.e. inspecting the behavior of the system during running becomes again an integral part of the authoring process. While first testing is more of a debugging character, final (second level) testing now means to explore the open system, to check if the goal to create openness was reached in a satisfying way.

## 6. Consequences for Interactive Storytelling

The introduction of independent objects with an own inner logic and the possible transfer of control away from the storygraph towards these objects gives rise to a *heterarchical* structure of the scenario. Authors are now obliged to find a balance for the potentially emerging concurrency between the entities at runtime. On the other hand we expect that the deliberate allowance for concurrency will lead to the creation of new forms of scenarios with the desired greater openness. Hence the authoring tool also facilitates the management of yet another category of concurrency which arises from the hierarchical structure of the graphs. Authors are able to

create event scripts in any node of any graph. But there is no automatism regarding inheritance of these scripts and encapsulated sub-graphs. The progress of a story may linger in a node for some time before maybe descending to a sub-node.

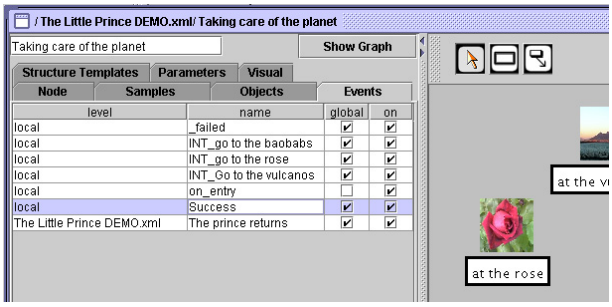


Fig. 7 Local and Global Scripting

At any hierarchical level an author now can decide whether or not to enable a script for substructures., i.e. make a script purely local or more global. This again may lead to (accepted) concurrency effects, e.g. between local and global timelines.

## 7. Summary

We have developed both a formalism and a prototypical authoring tool suitable for the ill-defined task of interactive storytelling or more general the creation of interactive scenarios. Compared to common software engineering processes we observe an additional level of openness for this task, namely the freedom of the user. Hence beside an approach to enable artists and authors to create such scenarios in the first place we also had to make sure that they can create these scenarios with the desired greater openness for the user. For this common formalisms like FSMs had to be extended. The resulting phenomena of concurrencies between local and global scripting as well as concurring heterarchical entities were deliberately accepted. We stressed the relevance of different levels of testing during all parts of the scenario development process. Authors with little or no programming skills can attend to the task of interactive scenario modeling and contribute their very own artistic skills.

## Acknowledgements

This work is part of the ongoing 'alVRed' project [1] and was supported by the German Ministry of Education and Research (BMBF) under grant No. 524-40001-01IRA06A within the 'Virtual and Augmented Reality' project framework.

## References

[1] Project *alVRed*, http://www.alvred.de

[2] MASA: *DirectIA*, http://www.directia.com

[3] Eastgate Systems Inc.: *Storyspace* http://www.eastgate.com/Storyspace.html

[4] Fischer, G.: *Shared Understanding, Informed Participation, and Social Creativity – Objectives for the Next Generation of Collaborative Systems,* in: Proceedings of COOP'2000, Sophia Antipolis, France, 20 00

[5] Fischer, G.: *Articulating the Task at Hand and Making Information Relevant to It,* in: Human-Computer Interaction Journal, Special Issue on Context-Aware Computing, (in press), 2001 www.cs.colorado.edu/~gerhard/papers/hci2001.pdf

[6] Harel, D.: *Statecharts: A Visual Formalism for Complex Systems,* Science of Computer Programming 8, Elsevier Science Publishers B.V. (North-Holland), 1987

[7] Mateas, M., Stern, A.: *Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade,* Technical Report CMU-CS-02-198, Carnegie Mellon University, 2002

[8] Screenplay Systems: *Dramatica* http://www.dramatica.com

[9] Stottler Henke, Inc.: *SimBionic* http://www.stottlerhenke.com/products/index.htm

[10] Wolfram, S.: *A New Kind of Science*, Wolfram Media Inc., Champaign, IL, 2002