Experiences in Designing a User-Oriented Tool for Building and Understanding Interactions in Virtual Environments

Cara Winterbottom*, James Gain and Edwin Blake

Collaborative Visual Computing Laboratory, Department of Computer Science, University of Cape Town, Private Bag, Rondebosch, 7701, Cape Town, South Africa

Corresponding Author - Tel: +27216502670; Fax: +27216899465
Email: cwinterb@cs.uct.ac.za

ABSTRACT (100-150 words)

Designing a virtual environment and its interactions is a difficult task because of the complexity of specifying non-deterministic relationships between multiple objects. We present a system to help novice designers create interactions in a virtual environment. Our system uses triggersets (event-condition-action triads) for entering interactions. It provides multiple visualizations of the virtual environment and its interactions: a sequence diagram for narrative sequencing, a floorplan for spatial sequencing and a timeline for time sequencing. We conducted an exploratory study with 11 subjects, where some received visualizations and triggersets of a VE and others only received the triggersets. The study had two parts: to assess whether subjects could sequence triggersets accurately and to assess how they managed to debug mistakes in a different set of triggersets. The visualization group described 72.5% of the sequence correct on average, compared to 56.4% by the non-visualization group. For debugging, the visualization group detected more than twice as many errors as the non-visualization group.

1. Introduction

Virtual Reality (VR) is no longer a novel medium. Its use has been extensively researched in many academic institutions and in a wide variety of projects. Research has varied from investigation of application areas, such as phobia treatment [1] and cultural heritage [2], to techniques for interaction with a Virtual Environment (VE) [3] and how to maximise the user's experience [4, 5]. However, to date it has only been used as a research tool or used for large-scale projects, such as 3D games. It has the potential to become easily and commonly used, like video or photography have become. However, in order for it to be more widely explored outside the academic context, it must be made more accessible.

VR is a specialisation of 3D world creation. It has all of the complexities of this and additional difficulties. These are introduced because VR presupposes an independent user. The creation of a VE has two main parts: static 3D content and dynamic interactions. Interactions are the relationships set up between the user of a VE, its objects and the environment itself. Creating the content is very like architectural and graphic design, where 3D shapes must be modelled, skinned and often given simple animations. The content must then be brought to life with the interactions, which are programmed or scripted. The design of interactions is difficult for several reasons.

- They happen over time for an indeterminate duration, so the design cannot be viewed statically.
- They happen for various entities, so that each entity or group of entities may have a different set of interactions in which they take part, which leads to a combinatorial escalation of possibilities for interaction.
- If there is a user, at least some of the interactions will be determined by what the user does, which cannot be pre-specified. Therefore, the interactions are non-deterministic and the designer must deal with a significant amount of uncertainty about how the end result will be experienced.
- Interactions include actions that are so commonplace to us that we do not even think about them, like keeping feet on floor, collisions, and avoiding obstacles in the space. They require a high level of detail to define.
- Very often the VE will have a purpose or tell a story, which means that the user must be guided by the interactions and the environment to achieve a goal.
- The VE must be sufficiently reactive to the user's interactions to make the experience enjoyable and interesting. This includes real-time reactions to user input.

Therefore, the expertise required both for designing and implementing a VE and its interactions puts VR out of the scope of many people, for whom creating a VE would be a small part of their work. For instance, a teacher might want to create a VE to illustrate a concept to her students, such as the effects of different amount of water, sunlight and minerals on various plants' growth. While she might have the hardware to run a desktop VE, she is unlikely to have the time and programming knowledge to create one.

We define people who may want to use VR, but for whom the skills involved are not part of their primary expertise, as content experts. As VR could be useful in many

fields, content experts will have primary expertise in a range of different disciplines, for example marketing, education, archaeology and psychology. They will have a wide variety of programming capability and design experience. This group is large and varied. Any system which attempts to help them in creating VEs must cater for those who cannot program and are novice designers, while still being useful for the others with more experience.

In this paper, we describe our process of researching and designing a system for content experts to conceptualise, implement and debug interactions in a VE, without having to explicitly learn a new set of skills. Because our target users are possibly novices in VE creation, we designed our system to assist them in building the requisite knowledge. The key means by which we accomplish this is in providing multiple visualizations of the VE and its interactions: a floorplan for spatial sequencing, timelines for time sequencing and a sequence diagram of the flow of interactions for narrative sequencing. Because the end product is so visual, we decided that it was important to be able to design with immediate visual feedback. We describe a study conducted to assess the effectiveness of our system, especially the use of visualizations for understanding and debugging interactions.

We begin by examining the approaches of other authoring systems in Section 2. This is followed in Section 3 by a discussion of theory and past research which influenced the design of our system, particularly in the areas of learning theory, visualization and VR design. In Section 4, our system is described in detail and in Section 5 we present our visualization study and its results. We conclude in Section 6 with a discussion of the implications of our work.

2. Approaches of other systems

We began our research by examining existing authoring systems aimed at novice programmers. We extracted three major axes along which we could analyse these systems: views provided of the authoring process, input style and feedback/ exploration/ error handling. Figure 1 shows how the axes combine when a user works with a system. The examined systems are summarised in Table 1, structured according to the type of authoring that they provide and the three axes mentioned above.

We wanted to examine a variety of authoring systems, so that we might learn from related fields. Therefore, the systems include authoring for dance, 2D simulations, 2D multimedia applications, 3D storytelling and a construction game, as well as three VE authoring systems. Commonalities in these systems led us to our axes for analysis. Most of them offered more than one view of the data so that end-users could author in different ways and gain different perspectives on the design. They all used simple mechanisms for input so that the end-user would not have to program, at least initially. They also provided various mechanisms for feedback and error handling, in attempts to assist the end-user and foster exploration.

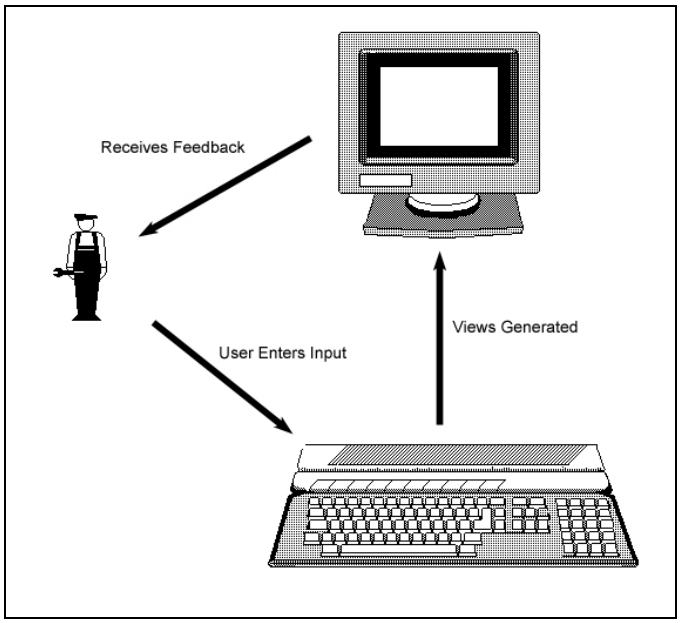

Figure 1: User enters input, which is translated by the system into views. These in turn provide feedback to the user.

A few notable points emerge from examining these systems. In terms of providing *multiple views*, none of the VR authoring systems provides more than one additional method of examining the interactions that are entered during the authoring process. In all cases, this is a 3D window. The 3D window is very important, as it shows what the end product will look like. However, novice users have been shown to have difficulties with understanding 3D interactions and how they are composed [6]. This would suggest that alternative aid should be provided to conceptualise interaction in the 3D space. In contrast, three of the five other systems provide more than one additional view. In particular, Anecdote [7], a system for authoring 2D multimedia applications, and Compose [8], a system for composing dance routines, provide five and three alternative views respectively. The views are specialised for different parts of the authoring process. For instance, in Anecdote, a scene view is provided for examining a single scene in detail, a link view is provided for examining how the scenes link together, a timeline view is provided for viewing a linear progression of scenes, an outline view is provided for a profile of all scenes with their file details and a cast scene is provided for viewing the content of the application. The Visions system [9] is the closest to VR authoring and it provides a timeline view, but it is not designed to create interactive, non-linear environments.

In terms of *input mechanisms*, all of the systems use some form of direct manipulation and drag-and-drop functionality. However, the VR authoring systems require that the end-user learn how to script or program in a high-level language in order to create more complex interactions. They assist end-users in creating VEs with simple interactions, such as architectural walkthroughs and collision-based reactions, using simple means. However, the possible interactions are limited to the basic building blocks for Virtools [10, 11], 3D actions that follow a 2D metaphor for VR-MOG [12] and simple rotation / translation / scaling actions for Alice [13,14]. Only Virtools allows 3D models to be imported with existing animations, although Alice does allow each part of a model to be

independently manipulated, which increases the functionality of the basic actions. Creating interactivity in Virtools requires programming in C++ or Python, or using the behaviour graph. End-users with no programming experience may have difficulty with this formalism, which requires the technical management of input and output parameters of building blocks.

In terms of *feedback and error handling*, all of the systems have some facilities to provide the end-user with immediate feedback and assist in the design process. Agentsheets [15,16] and SimCity [17] in particular provide tools for querying the functionality of each part of the system and for stepping through the output. The VR authoring systems, except for Virtools provide limited support for error handling and debugging. The drag-and-drop sentence functions of Alice stop end-users from programming incorrect syntax, but there is no provision for interrogating the semantics of the input. Virtools provides fairly extensive debugging facilities, but these are similar to tools provided for high-level programming languages (trace feature and breakpoints). Such tools have been shown to be of limited use even to experienced programmers [18].

System	**Type of Authoring**	**Multiple Views**	**Input Style**	**Feedback/ Exploration/ Error Handling**
Agentsheets [15, 16]	2D Agent-based Interactive Simulation	2D grid shows effects of rules throughout authoring process, while rules are composed using drag-and-drop icons and sentence functions	Direct manipulation; Drag-and-drop from palettes; Rule-based	Conditions, actions and rules can be tested without constructing whole program; Explain button describes what elements do; Can step through simulation
Alice [13, 14]	Virtual Environments	The 3D window for viewing interactions and manipulating objects is the only way to view the programmed interactions, apart from the drag-and-drop sentence functions	Drag-and-drop; Textual edit boxes; Direct manipulation, but more complex behaviour must be coded with Python or C++	Uses non-mathematical terminology for translation, rotation and scaling; Unlimited undo
Anecdote [7]	Multimedia 2D Applications	Five views for editing: scene, link (flowchart), timeline, outline and cast (contents)	Drag-and-drop; Text; Sketching	Uses surrogate media to represent content for early feedback

System	Type of Authoring	Multiple Views	Input Style	Feedback/ Exploration/ Error Handling
Compose [8]	Dance Composition	Stage for spatial relationships; Timeline uses a score with row for each figure to change from beat to beat; Body screen for creating body positions in 3D	Drag-and-drop from menu of postures and sequences; Direct manipulation	Animates final composition for evaluation; Flexible movement between levels of abstraction and views
SimCity [17]	Simulation Game	No - But main map can be changed to view different aspects of construction, and can be zoomed	Tool-based; Drag-and-drop	Information bar shows colour-coded current information about construction; Navigation map for orientation when main map zoomed; Drop shadow; Query tool for information on elements; Simulations run at various speeds with pausing facilities
Virtools [10, 11]	Virtual Environments	The 3D window for placing and manipulating objects is the only way to view the programmed interactions apart from the behaviour graph (like a state chart), which is used for authoring by connecting behaviour blocks and objects	Direct manipulation; textual edit boxes; Drag-and-drop scene manager and behaviour graph for interactions works for low-level behaviours, but more complex behaviours must be coded using the SDK; 3D window provides tools for manipulating entities	Large library of behaviour building blocks; Unless coding is performed, no recompilation is needed so feedback is immediate; While playing, a Trace feature shows which behaviour blocks are activated and their values; Breakpoints to pause execution; Graph for interaction input makes workflow easy
Visions [9]	3D Storytelling Applications	Plug ins for timeline editor grouped in rows by object then by type of event, camera editor (3D view), 3D set modeller and character generation	Direct manipulation in 3D view	Manipulators allow for visual rotation, translation and scaling of objects; Library of objects

System	Type of Authoring	Multiple Views	Input Style	Feedback/ Exploration/ Error Handling
VR-MOG [12]	Virtual Environments	No	Direct manipulation; Textual edit boxes; Predefined 3D widgets with links to callbacks; Interactions not based on 2D metaphor must be coded	Built on extension of 3D model of interaction with which users are familiar; 2D floorplan mimics the final UI for immediate feedback

Table 1: Survey of authoring systems structured by the style of authoring, provision of multiple views, input mechanisms and methods of feedback and error handling

Another interesting point about these systems is that, apart from Alice, they do not provide experimental evidence of how they can be used and by whom. Therefore, while many of their features are probably useful, there is no data available about how end-users work with them.

3. Theoretical Underpinnings

In this section, we discuss areas of research that were influential in our own research. We consider three broad areas of research: learning theories and user interface design guidelines; the use of visualizations to support design and debugging; and VE authoring. A trend that runs through all three areas and that will be highlighted in each is the attempt to support novices and learn from expert practices.

3.1 Learning theory and user interface guidelines

Because our target users are possibly novices, our system can be considered a learning environment, where we guide users towards good VR design practices. Research by Ko et al. on end-user programming systems has indicated that several aspects of tools can act as learning barriers [19]. In other words, these are aspects of systems which foster incorrect assumptions on the part of a novice user about how to proceed. The authors came up with several guidelines for designing more learnable systems, such as providing scaffolding to foster creativity for overcoming design difficulties and making the rules of the system clear. Blackwell [21] has defined the attention-investment model, where a learner will weigh the cost and benefits of overcoming barriers, and may decide to abandon the tool if the risks outweigh the rewards.

In order to minimize learning barriers, we examined educational advice. A theory that has been applied extensively in education to support learners in actively building their knowledge is constructivism [21, 22]. Constructivism is a theory of the structure of knowledge and complexity, where knowledge is believed to be constructed by individuals through interaction with their world [23]. The theory describes various ways

of handling the complexity of knowledge creation. It provides principles to guide the learning process. These are as follows:

- The learner should be provided with multiple perspectives on a problem and multiple representations of it. Multiple perspectives allow complexity to be broken down by focussing on different aspects of it. This in turn allows a deeper understanding when the distinct representations are understood as describing facets of the same idea [25]. Several authoring systems have successfully used this principle to support their work process. For example, in Compose [8], alternate representations are available so that they can be juxtaposed and the composer can think in different ways about the composition. Baldonado et al [25] provide guidelines for the use of multiple views in information visualization. The authors state that they should be used when there is diversity of information, when different views elicit correlations or disparity in the information, or when complex data can be decomposed to manageable chunks.
- Each piece of new information should be kept as simple as possible. The complexity of knowledge can be built up through links between information chunks. This makes new information easier to assimilate and provides incremental building of knowledge. For example, Agentsheets [15, 16] uses a rule-based system where complex simulations can be built up through combining rules, which consist of simple triggers, conditions and actions. These can be checked individually before the whole method is composed.
- Exploration should be encouraged and supported through scaffolding [26]. Systems should be forgiving of mistakes, as these are part of the learning process. Errors should be easily identified by the learner and informative help should be provided in recovering from them, so that they do not become learning barriers [19]. Part of this can be achieved through providing immediate and useful feedback about the user's actions in the system. Development time should be shortened, or at least allow for prototypes to be developed quickly, so that the designer gains positive feedback for further exploration. All of the VR systems examined in Section 2 allow for early prototyping and provide some facilities for scaffolding the learning process. For instance, VR-MOG [12] provides a 2D metaphor so that learners can use existing knowledge about 2D interactions.
- The learner should feel in control of the process, which means that systems must make clear how the programming relates to the effects that are achieved. There should also be a large amount of flexibility in the tool so that the learner has freedom of expression. This should encourage the learner to reflect on the process and connect the new information to existing knowledge structures. For example, in Anecdote [7], multiple views provide support for various authoring styles (graphical/textual, linear/non-linear) at different levels of the design process. This allows learners to use the design style that works best for them.

These principles tie in with general 2D user interface advice. Norman [27] describes the gulfs of execution and evaluation. The gulf of execution refers to the disparity between what the user wants to do and what is allowed, and the gulf of evaluation refers to the disparity between what the user expects to happen after an action and the system's representation. He specifies four principles of good design to combat these gulfs: system state and action alternatives should be visible; there should be a good conceptual model with consistent system image so users can predict effects of their actions; the

interface should have good mappings which reveal the relationships between stages of action, between the interface and what happens underneath and between actions and their effects; and the user should receive continuous feedback. In addition, errors should be easy to detect and have minimal consequences to encourage people to explore. Shneiderman echoes Norman's advice and states that an effective system will engender feelings of accomplishment, control and clarity in its users [28].

3.2 Use of visualizations to support design and debugging

Visualization research has a long history, particularly in scientific visualization and the visualization of large information datasets [29, 30]. In recent years, research has increasingly been conducted into the use of visualizations for an even greater variety of problems. The use of visualizations to support programming tasks, such as debugging and control structure creation, especially with novice programmers has been explored [31, 32, 33]. According to Romero et al. [31], program visualization aims to find a way to help the designer see the flow of control through a program – what is happening and potentially what unforeseen events may occur. Pane [34], in research conducted on novice programming systems, found that providing different ways of representing information was often useful for different tasks involving that information, for example coding and parsing the code. He also found that: "Experts are more likely than novices to develop complex high-level representations of the program. This happens in a top-down fashion when looking at large units and fitting large pieces together, and in a bottom-up fashion when identifying chunks of code and deducing how they fit into the goal hierarchy … Anything that the environment can do to assist novices in forming high-level representations may be helpful." [34, p. 275]

Myers et al. [35] have recently researched how to make programming environments more natural for end-users. As part of this research, they studied the language and style that non-programmers use to solve problems. These studies elicited some interesting results: subjects naturally used an event-based style; they were often confused by Boolean expressions; and they often sketched the layout of the problem solution, but used text to describe events and actions. As part of this research and as a response to a lack of useful debugging tools, Ko and Myers developed a methodology for analysing programming errors [18]. They also developed Whyline, an interface to support debugging activities [36]. This tool was tested with novices and experts, and found to be used successfully by both. It supports the debugging task by providing visualizations of a program's runtime in response to questions about what went wrong.

Research has also been conducted into how visualizations are used by experts during various design processes. Petre and Blackwell [37] studied the mental imagery employed by expert programmers during program design, using observation and interviews. They found that experts all used various forms of mental imagery to think about a task, although verbal discussion was also important. Commonalities in the imagery used included the fact that they were all dynamic, but could be stopped and reversed; they had adjustable granularity; they could incorporate fuzziness where thinking was incomplete; images were often labelled; they all included simultaneous multiple images; and the mental images used were related to the problem-space, rather than the programming paradigm. Research conducted on how architectural experts use sketches while designing [38, 39] suggests that drawings are essential to the design

process. Experts revisit their sketches throughout the design process, using them as a reference for thinking and reasoning about their designs. General research on external representations suggests that they are useful in promoting reflexivity and a deeper understanding of the subject of the representations [38, 40, 41, 30]. Eastman [41], in a survey of representations used in design, suggests that novices learn from viewing and working with external representations, so that in time these are internalised and part of the designer's reasoning tools.

Blackwell et al. [42] examined the field of software visualization and concluded that there were many unanswered questions. For instance, for which problems are diagrams better than text, how do experts work with diagrams, how do individuals differ in how they work with diagrams and what are the benefits of using multiple representations?

3.3 VE authoring research

In research into VE design and authoring, many guidelines have been provided for the design of successful VEs, both in terms of the design process and in terms of the qualities of a successful VE [43, 44]. However, these are have not been integrated into tools for assisting the novice designer.

Two studies recently conducted into how non-programmers and novice designers work to create VEs helped us to understand the needs of our target audience.
In the first study, experimenters worked with students of an undergraduate course, which was an introduction to VE design and development: by the end students had to design and develop an interactive 3D game [45]. None of the students had programming experience. They were provided with a workshop on interactions and how to specify them, and given a simple programming tutorial using Alice [13,14]. The experimenters observed how students represented interaction ideas and their process in designing the games. Students provided floor plans, flow diagrams and pseudo code to the experimenters. Some insights from this experience were that subjects produced very linear designs and described interactions in terms of a 'winning walkthrough'; subjects naturally used floorplans and flowcharts to show the interactions they were designing for the game; and subjects had considerable difficulty with programming constructs, even while using Alice, which is a novice programming system. However, they were not given many contact hours with Alice in order to become familiar with the concepts.

In the second study, the researcher worked with content experts from various application areas, who had managed to use VR authoring tools to create applications [46]. He collaborated with these experts to discover the difficulties and problems with their tools. The interaction-related problems that were identified were as follows:

a. Multiple sequential tools for authoring – Subjects did not like the fact that with most tools one task could not be started until another was completed. They wanted to be able to author VEs as they chose, rather than complete steps in a required order.
b. Compile and view model – Subjects found it hard to mentally translate between different representations before and after compilation. They also found the lack of immediate feedback disconcerting.

c. Interaction specification interface – The tools for specifying interactions mostly required extensive programming knowledge which was agreed to be one of the most difficult tasks in VE authoring. Subjects would avoid defining complex interactions because it was too daunting.
d. Time manipulation – The facilities for dealing with time, in terms of authoring and viewing the created VE led to a lot of time wasting due to lengthy content. Subjects also had difficulty previewing simultaneous actions. Creating time-based sequences was also seen as very difficult.

We addressed these issues in the design of our tool, by applying the learning theory and 2D user interface advice discussed in Section 3.1. In terms of (a), we referred to the principles of user control and exploration and ensured that designers could author a VE using their own style. Workflow was designed to be as flexible as possible, with no required order of tasks. In terms of (b), by applying the principles of multiplicity and exploration, we provided immediate feedback using 2D visualizations of the interactions. We also hoped that by providing multiple visualizations, we could help novice designers to think in non-linear terms about their interactions. In terms of (c), we tried to allow designers to create complex interactions by combining simple parts, by using an event-action interface. In terms of (d), we provided a timeline visualization, which allows designers to skip through lengthy time sequences and place actions accurately in time. In the next section, we discuss our system in more detail.

4. VE Interaction Authoring System

In this section we describe our system in detail. There are various aspects to consider in VE interaction design: the use of space, time and possible sequencing of interactions – in addition to how the user will move. Our visualizations provide an idea of the VE's space and how it is used during the interactions. They also provide an idea of how time progresses in the VE. This cannot all be accomplished with one visualization, so multiple visualizations are provided to which the designer can refer as he or she wishes. The use of multiple visualizations of a problem also allows the complexity to be broken down by focussing on different aspects of it. Three visualizations are used to assist the design and debugging process: timelines, which are both a construction and a visualization tool; a floorplan, which provides a basic visualization of the space of the VE; and a sequence diagram which is most useful for debugging the complexity of the interactions, and to understand why triggersets execute or not. These visualizations provide the benefits of a visual system, such as an overview of the data and different perspectives on it. They help the designer to conceptualise and plan complex relations among the objects of the VE.

Although much of the system is visual, we decided not to use a visual language for the programming. This means that we do not require learners to use a new visual syntax. In addition, as mentioned in Section 3.2, studies have shown that people naturally specify computer behaviour using text and an event-based style [35]. While VE interaction implementation of a certain complexity is usually accomplished through programming with high-level languages, it is very simple: if these conditions apply then this happens; while that is happening, this is also happening, etc. We therefore decided to base the interactions on an event-based language that is programmed using dialog boxes and text. The event-action paradigm was chosen as this allows for simple events, conditions

and actions to be specified, which can be combined to form complex interactions. We use dialog boxes for the input of information as people in general are comfortable working with packages such as Word, Excel and Wordperfect, which often use dialog based input. Dialogs provide constraints which guide users in the right direction, but they are also flexible. We designed our dialogs using with Shneiderman's golden rules of dialog design [28]. These include providing consistency and closure, informative feedback, simple error handling, easy reversal of actions, user control and simple displays. Figure 2 shows how the designer interacts with the VE world, the triggersets and the visualizations.

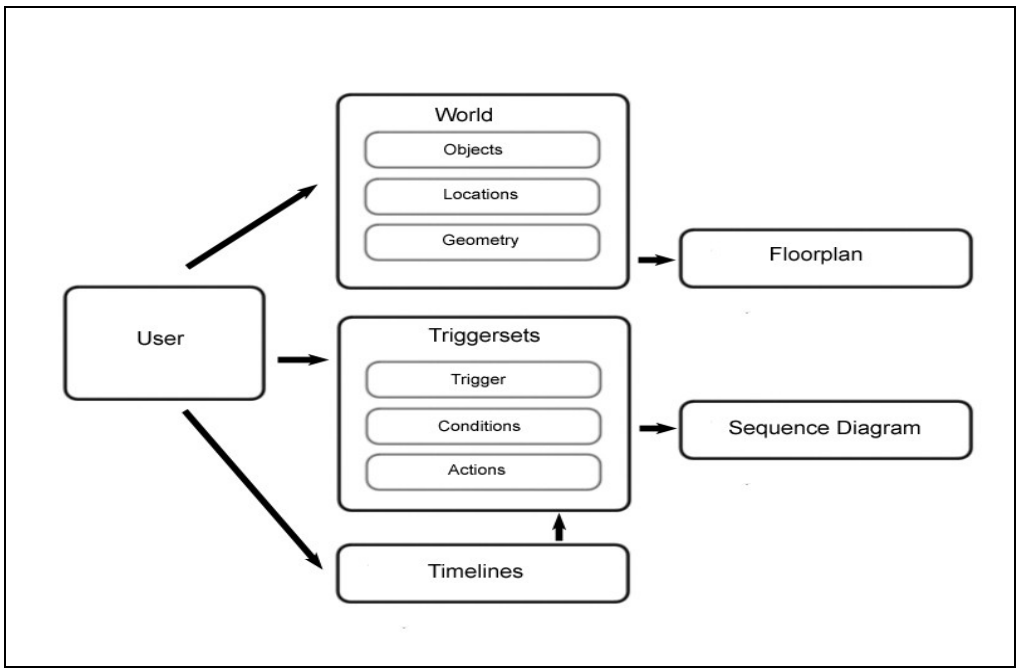

Figure 2: The user enters triggersets, 3D object details and timeline information into the system. The timelines may be added to the triggersets. The sequence diagram is generated from the triggersets and the floorplan is generated from the 3D world details.

The process of using the system is as follows: The designer specifies objects and the environment. These do not have to be instantiated in their 3D forms immediately; only their coordinates, names and approximate details (such as sounds and animations) must be specified. For programming the interactions, we used an event-action paradigm. Trigger-condition-action triads, which we refer to as triggersets, were used. These are described in Section 4.1, but are essentially rules for specifying how the interactions will happen. Time based sequences of actions can be entered using timelines, which are described further in Section 4.4. The system generates a floorplan from the object positions and locations in the environment that have been specified. Floorplans are described in Section 4.2. The system also generates a sequence diagram from the triggersets that have been created. Sequence diagrams are described in Section 4.3. As soon as the designer has entered a few triggersets and object details into the system, she can view the sequence generated by these interactions and how objects are arranged on the floorplan. Therefore, the system allows for incremental programming, as the status

and consequences of what has been programmed can be checked at any point by examining the visualizations.

The event-action input mechanism is enriched by the addition of the diagrams, which help the designer to disentangle and debug the entered interactions. Unfortunately, research indicates that external representations are most effective when drawn by the designer [40, 38, 39]. This is not possible here, as the visualizations are provided as scaffolding for the designer in the complex task of creating interactions. While some of the target users might be able to create their own diagrams to describe the VE and interactions, many do not have the knowledge or skill to do so. Therefore, we added the ability to interact with the visualizations so that designers can make the visualizations their own and feel in control of the process.

4.1 Triggersets

The system for entering interactions provides for easy input of interactions in a dialog-based fashion, where combinations of simple actions provide for much power and flexibility [47]. The different triggers that are allowed were carefully selected, so that, with a few simple combinable options, complex interactions could be set up. These were selected based on previous experience of the kinds of interactions that are available in 3D games and that have been programmed in VEs.

As mentioned above, event-condition-action triads named triggersets are used to set up interaction in the environment. Figure 3 shows the format of the triggersets. Interactions are event-based, which means that anything that happens has to follow from an event. Triggersets can be activated by the user moving around and using the keyboard, or as a consequence of previously triggered actions. A triggerset consists of one triggering event, one or more conditions that are required for it to execute, and one or more resulting actions. Actions, triggers and conditions are all separate objects. Any combination of triggers, conditions and actions can be put together. Conditions can be combined with boolean AND. We decided not to allow boolean OR as a combination mechanism, as it has been shown that novices have problems with deciphering complex combinations of boolean logic [35]. Designers can mimic boolean OR by creating multiple triggersets.

The triggersets can be set up to introduce contradictions, but these are made apparent from the visualizations. For instance, a trigger that never fires will not appear on the sequence diagram and the floorplan will show where objects will collide with geometry. The principles of exploration and error-handling suggest that designers should be allowed to make mistakes so that they learn from them, but that they should be provided with feedback about the mistakes and assisted in recovering from them.

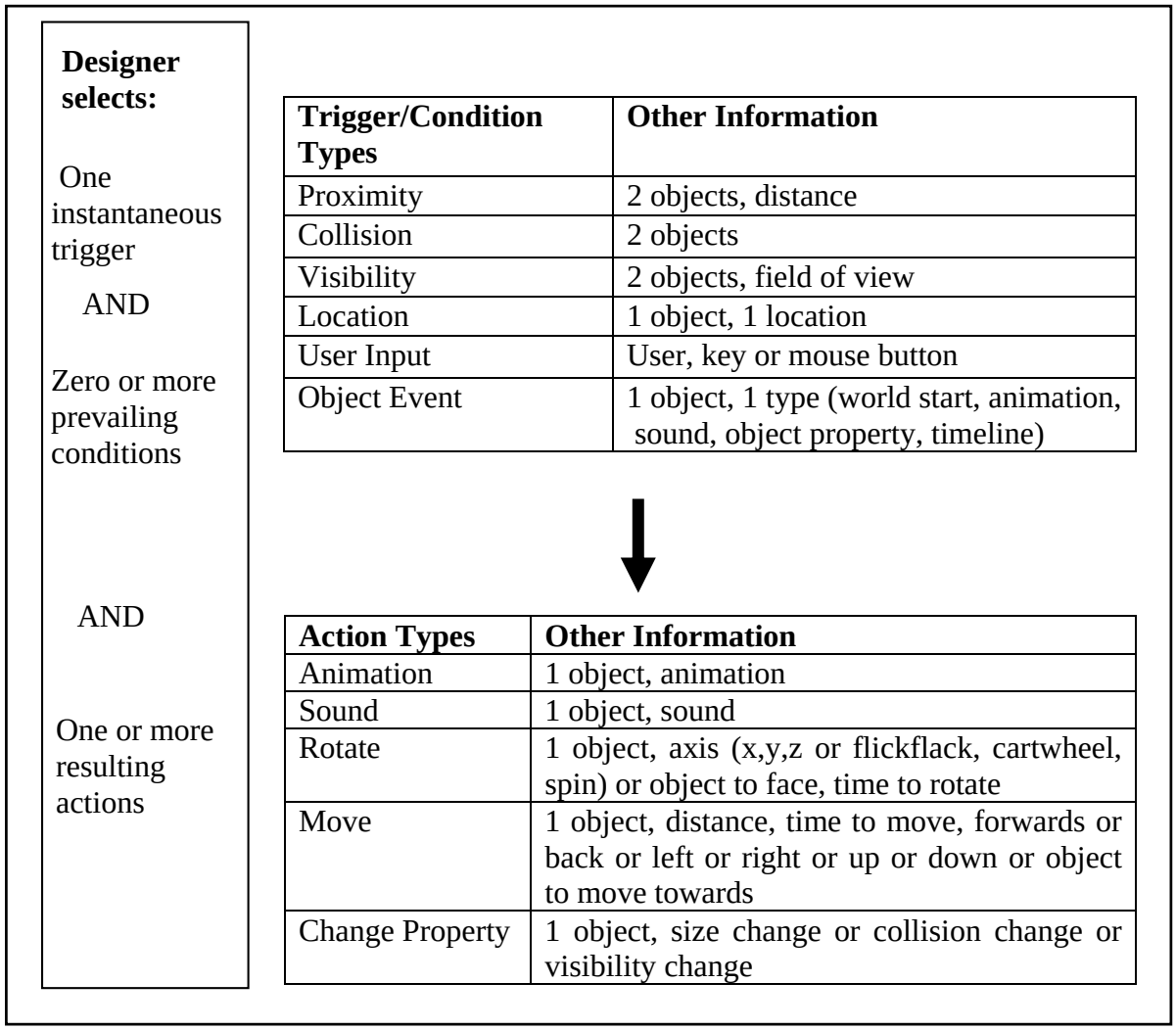

Figure 3: Diagram of the triggersets, showing how the designer selects one instantaneous trigger, zero or more conditions and one or more actions to create a triggerset. The kinds of triggers, conditions and actions that are available are also indicated.

Figure 4 displays the user interface of the main window of the system and Table 2 displays some examples of typical triggersets. For the user interface design of the triggersets, several paper mock-ups were produced based on user interface principles [27, 28]. Heuristic analysis was performed on them in several iterations.

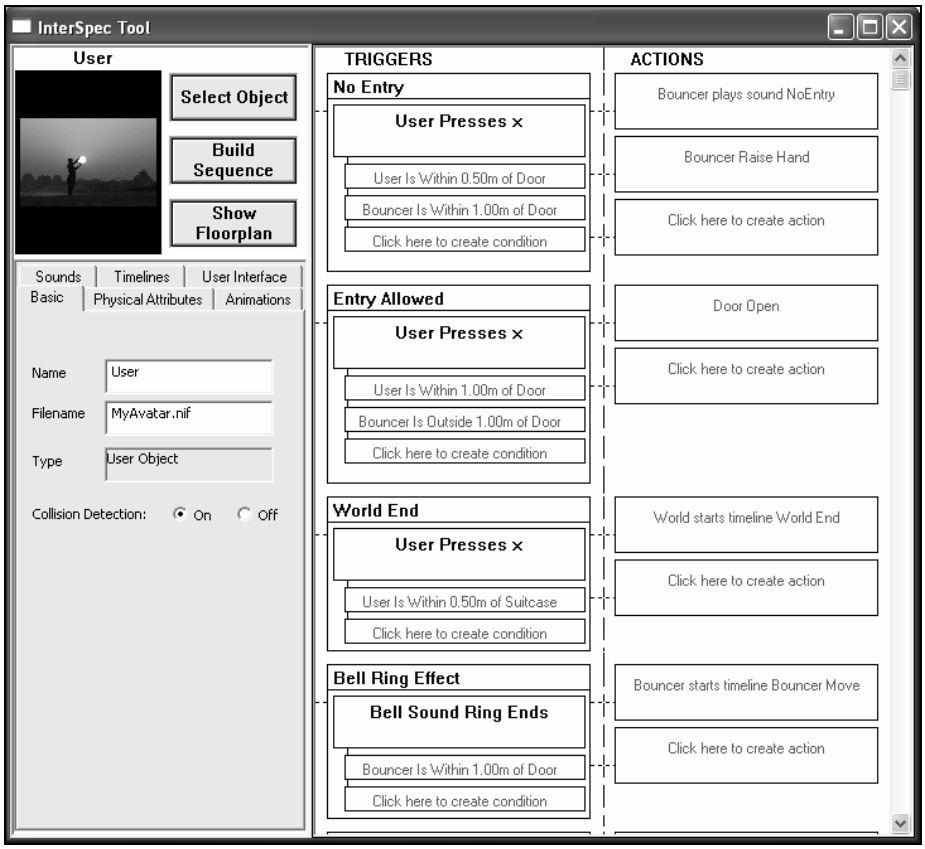

Figure 4: User interface for entering and examing triggersets and object details.

	TRIGGER	CONDITION(S)	ACTION(S)
Type	user input	proximity	play sound
Example	User presses ‘a’	Tom is within 1m of Table	Tom plays sound ‘You’re stuck here’
Type	object event	location	rotation AND movement
Example	Tom animation ‘shuffle’ ends	Sarah is in Lounge	Sarah turns to face Door in 10 sec AND Tom moves forwards 6m in 3 sec
Type	object event	user input AND visibility	start timeline
Example	World starts	User is pressing ‘z’ AND User is looking at Sarah	Sarah starts timeline ‘Ask for help’

Table 2: Examples of typical triggersets. The last three columns display triggers, conditions and actions respectively. Each row displays the type of trigger, condition or action and an example.

4.2 Floorplan

The floorplan has been shown by our own and other experiments to be very useful in the design of 3D worlds [45, 12]. Floorplans are also used successfully in engineering and architecture to represent space, e.g., CAD packages. Floorplans are maps, which most people use very easily [30, 48] and they allow complex 3D worlds to be viewed in a more simple 2D way. Fencott [49] provides a framework for designing a VE, using different design stages. One of the stages involves thinking about the perceptual opportunities of your VE. These create a narrative path through the VE, encouraging the user to explore the VE correctly. The floorplan helps the designer to create a perceptual map of these opportunities by providing a simple 2D visualization of the space in the VE. In particular, lines of sight, locations and positions of objects can be easily viewed.

In our floorplan, the space may be divided into rectangular spaces called *locations*, which can be used as triggers or conditions without specific coordinates. The floorplan automatically displays the positioning of any object that has been given spatial coordinates and indicates its orientation. If any proximity triggers have been set up, these are shown on the floorplan. The floorplan is marked with a grid according to the units of the world, so that the size of the space can be easily interpreted. A compass is also shown, to indicate direction in degrees and to assist the designer in working out the extent of an object's rotation.

Designers can interact with the floorplan using direct manipulation. They can select objects that are displayed. When an object is selected, it is highlighted and its proximity triggers, name and facing are shown. In future work we will allow objects to be positioned using direct manipulation, via the floorplan. The floorplan can also be layered to reduce complexity and to show different levels of a 3D world, if necessary. Figure 4 is an example of a floorplan used in the system.

4.3 Sequence Diagram

The sequence diagram is generated directly from the triggersets that have been entered by the designer. It follows the flow of data through the program, based on the triggersets. From this, states can be identified where specific interaction possibilities exist (i.e., where user interactions will have consequences). Each state specifies the current conditions in the environment that might allow triggersets to execute. The states are linked by arrows, which correspond to triggersets executing and leading to new states. If a triggerset does not change the state of the environment in a way that will allow more interactions to happen, it leads back to the same state.

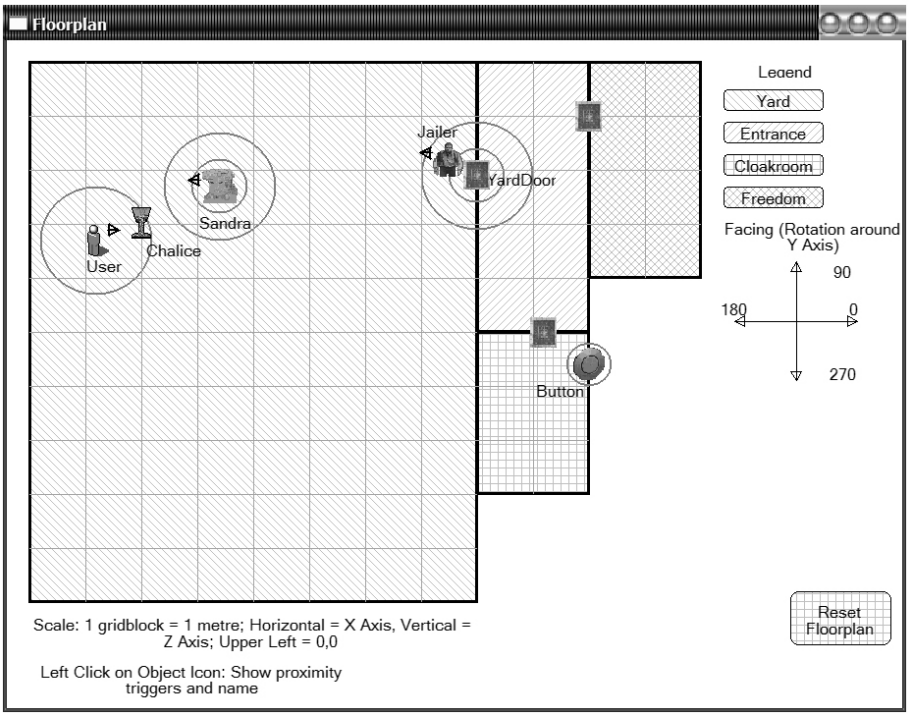

Figure 4: Floorplan showing locations, positions and orientations of objects, and proximity triggers.

The sequence diagram allows movement from individual interaction to a sequence. Thus, the sequence diagram can be seen as visualising narrative sequencing of the VE. The VE narrative sequence usually depends on the user's actions, so it can also be seen as showing possible reactions of the VE to the user. These reactions all have to be programmed into the system, so there are a limited number of possibilities. For larger sequence diagrams the ability to zoom and encapsulate will be added. This is in keeping with the visualization mantra: overview, zoom and filter details on demand [29]. The diagrams are inspired by Harel's statecharts [50, 51], which were developed for designing and maintaining complex reactive systems. These are based on reactions to discrete occurrences and allow complex information to be viewed in a manageable way. In a VE the discrete occurrences may be seen as the triggering events, which can have various consequences depending on time, space and state.

Designers can interact with the sequence diagram: by clicking on a state, those to which it leads are highlighted, as are the arrows leading to them; by clicking on an arrow, all other arrows of the same kind (i.e., referring to the same triggerset) are highlighted. In this way, designers can step through their interactions. In terms of debugging interactions, the sequence diagram provides designers with a visual representation of their interactions. They can see the effects of the triggersets that they have created, where triggersets have unexpected consequences and which triggersets never execute. Because the sequence diagram is not linear, designers are encouraged to view the interactions in a non-linear way. A sparse diagram will also indicate a lack of interactions provided in the VE. Superstates can be used to manage complexity for larger sequences. For example, a timeline is a superstate, where new interactions are only possible when timeline ends, but individual actions on the timeline may have different consequences. Thus, a timeline state may be opened to view the individual

actions and their consequences. Figure 5 is an example of a sequence diagram used in the system.

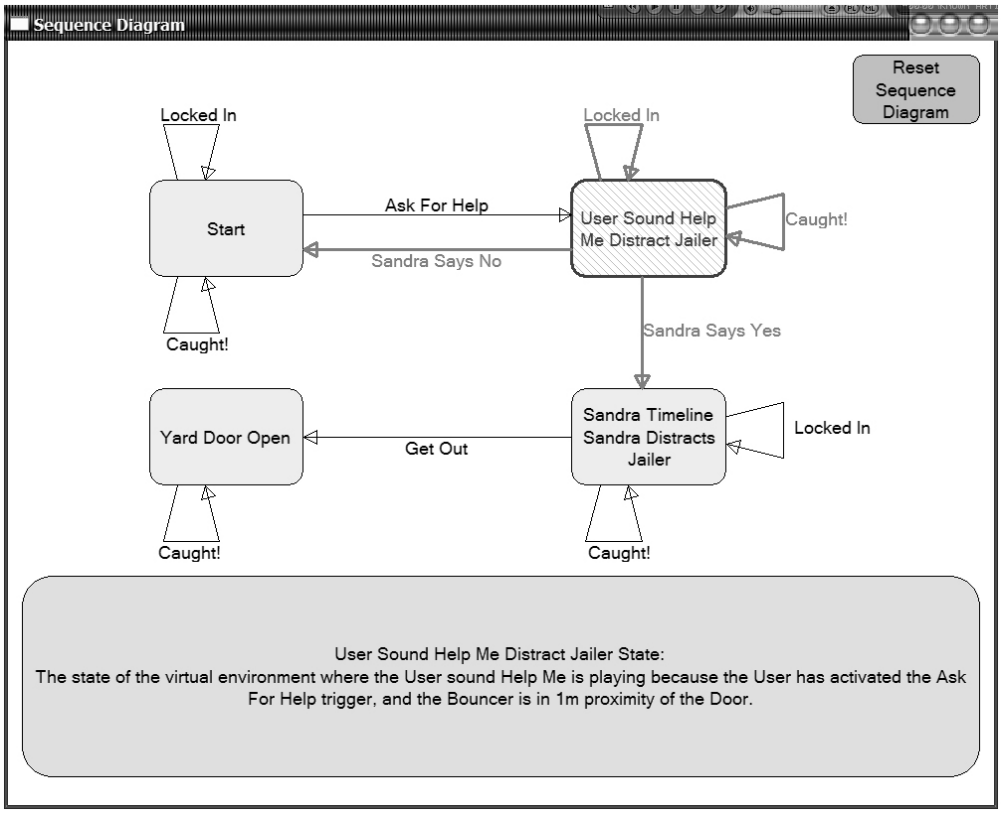

Figure 5: Sequence Diagram showing states and links. A state is a point from which user interactions will have consequences, and a link is a triggerset that can be executed from its originating state. Highlighting of one state and triggers leading from it are also shown. A description of the highlighted state is provided at the bottom of the diagram.

4.4 Timeline

Timelines are provided for predictable sequences of actions that will happen in known time. These can be started like any other action. They are also added to the triggers and conditions. Therefore, actions can be triggered by a timeline starting or ending, and a timeline happening or not can be specified as a condition. The timeline is divided hierarchically according to the objects that are involved. Thereafter, each row contains any actions that the object performs, specified according to starting time. The length of the action is automatically calculated from the action set up (e.g., an animation's basic length is read from its file and then multiplied by repetitions) and visualised on the timeline. Any action on a timeline can be selected to uncover more details about it. This opens the same dialog box that was used to create the action. Visually, the timelines can be used to review how the objects and their actions interact. Timelines are a very well understood formalism and have been shown to reduce errors in temporal ordering [7, 9]. We decided to use simple, non-branching timelines, as this reduces the complexity of each timeline [52]. Multiple timelines can be used to specify multiple series of events. Since timelines only display predictable sequences of events, no actions by the user can

be recorded on a timeline. If a possible user action introduces a branch, the current timeline will end. A new timeline can be started by the same or a different action. For example, if a group of actors are conversing and stop when the user approaches, the conversation can be captured on a timeline. When the user enters proximity of the actors, a triggerset can be defined which will stop the conversation timeline. If the actors perform two different sets of actions depending on the position of the user, this can be captured with two triggersets. Each will be triggered by a user position and start a different timeline. From examining the triggersets and sequence diagram, it will be obvious that the user actions have two different consequences. Figure 6 displays a typical timeline from the system.

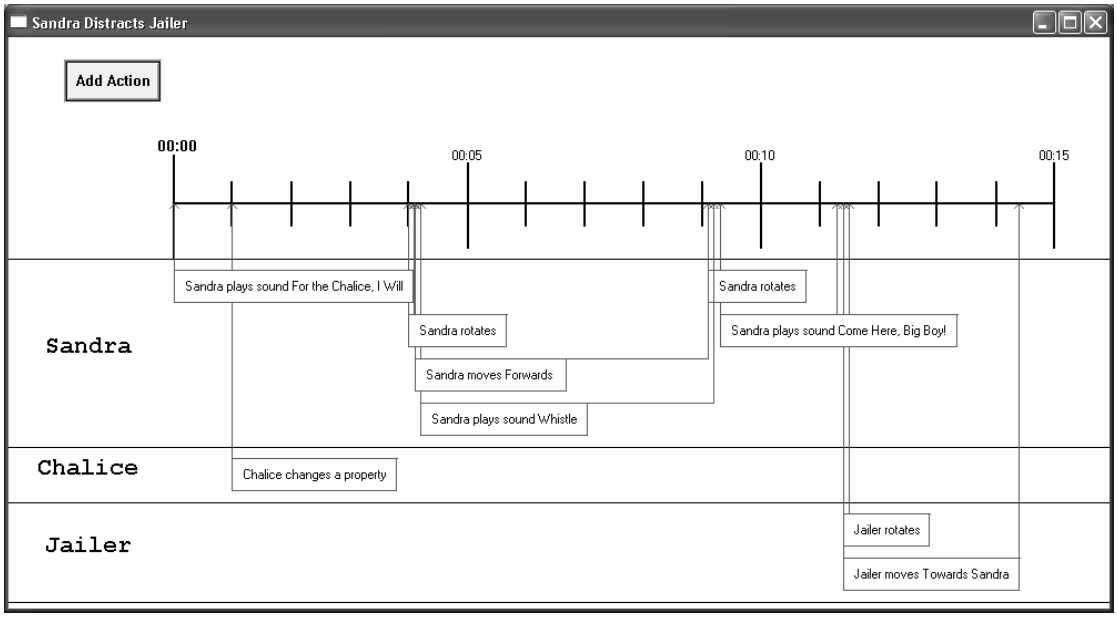

Figure 6: Timeline showing objects involved and their actions. These can be clicked on to elicit more detail about each action.

4.5 Linking between visualizations

All of the visualizations described are linked to each other. The objects and locations referred to in each visualization connect them to each other and to the triggersets, so that they can be cross-referenced. Timelines are described in the sequence diagram. For example, Figure 5 displays a sequence diagram for a VE where the state, *Sandra Timeline Sandra Distracts Jailer*, describes the state of the VE while the timeline, *Sandra Distracts Jailer* is playing. From this diagram, one can see that the triggerset *Sandra Says Yes* begins the timeline and that three triggersets can be activated by the timeline or actions contained in it. If the designer selects this state, a detailed description of the timeline is provided and if the designer selects any of the triggersets entering or leaving the state, a detailed description of each is given. The designer can also use the description of the state to find and open the actual timeline (Figure 6), from which more specific details of the actions on the timeline can be viewed. In this way, by working through the visualizations together, the designer can gain an overview of the sequence of triggersets and how they interact with each other, as well as a detailed description of each.

The visualizations were designed to be used together and to complement each other, so that the designer can become aware of their conceptual relationship: how space, time and sequencing affect each other. Most important are the sequence diagram and the floorplan, as these provide a view of the entire environment. When the 3D window is added to the system, a run mode will be developed, where a highlight moves through all of the visualizations indicating where the action is taking place. This should encourage reflection as the visualizations can all be viewed simultaneously. It will help the designer to step back and think about what the user may do, based on the diagrams in VE world terms, not in terms of the programming.

5. Visualization Study

5.1 Aims of the study

An exploratory study was conducted to test the effects of the visualizations and the triggersets in our system. The visualizations were prototypes, without some of the functionality that would later be added. For this initial study we wanted qualitative information about what people preferred to use and about their process for working. More specifically, we wanted to study how effectively the visualizations and triggersets were used by people who did not work in computer graphics, in understanding a sequence of events and debugging errors. This general aim can be subdivided:

- To test the user interface of the triggersets and assess how well people work with them, in terms of understanding what they mean and their implications.
- To observe people using the provided visualizations: timeline, floorplan and sequence diagram. To see which they use and how they use them. To discover how people work with multiple windows. To see if people could understand the visualizations.
- To assess whether people who experienced the visualizations would be able to describe the possible sequence of interactions in a VE more accurately than people who did not have the visualizations to help them. We also wanted to test whether they would be able to more accurately identify what was wrong with a second set of interactions.

5.2 Methodology and framework

Because of the exploratory nature of our work, we decided to use ethnographic techniques to conduct the study [53, 54]. Therefore, we observed subjects working on simple problems using our system. The subjects were observed in informal surroundings, so that they would feel comfortable. After the tasks were finished, the researcher discussed the subjects' experiences with them. This was structured according to specific interview questions, but was conversational in form so that subjects would feel comfortable giving their honest impressions of the system.

As mentioned above, part of the study was about identifying incorrectly entered interactions. For this section, knowledge of typical mistakes in VE authoring came from the studies described in Section 3.3. and experience of members of our computer graphics lab over three years of VE programming. Most mistakes in our experience fall

into one or more of four general categories: timing, spatial, sequencing/logical and implicit assumptions. These are described in Table 3.

Category	Description	Example
Timing	Errors arising from the time it takes the user or other objects to complete actions	User does not have enough time to get through a door before it closes
Spatial	Errors concerned with the way space is used, in terms of orientation and location of objects	An object is set to turn the wrong way and therefore moves in the wrong direction
Sequencing/Logical	Errors in the ordering of the triggersets and the way that they interact	A trigger never executes because it is not accessed by other triggers
Implicit Assumption	Forgetting to state all behaviour explicitly	Designer assumes that an actor is facing the User

Table 3: Typical Interaction Programming Mistakes: Timing, Spatial, Sequencing/Logical and Implicit Assumption. The columns show the category of mistake, a description of the category, and an example respectively.

The fact that mistakes often fall into more than one category suggests that having multiple perspectives on the debugging process will make it easier to find them. If a mistake is not found by examining one aspect of the program, it might be found by examining others. Timing is the most difficult error-type to detect. It is almost always a user-related problem, which means that the designer has to execute and step through the VE to be able to understand whether and when there will be a problem. Run-time debugging is part of the future work intended for this system.

5.3 Subjects

A sample of 11 post-graduate students and working people from different disciplines was recruited. Table 4 lists subjects' gender, occupation and self-rated programming and graphics experience. It was decided to use people with postgraduate degrees or equivalent working experience, as these corresponded to the target group. All subjects were experienced at working with computers, as we did not want lack of familiarity with the use of computers to be an extraneous variable in the study. The disciplines were selected to cover a wide range of target users. We recruited people with a range of programming experience, as many target users have done a certain amount of programming. None of the subjects had any experience with graphics programming, although they had a range of experience with graphics packages. Apart from the computer science subjects, all of the programming was of a basic level, such as flash, Visual Basic and Matlab. We recruited computer science subjects, as we wanted to see how useful these subjects found the visualizations and how they worked with the triggersets. Subjects were volunteers within these constraints and were paid a small amount for their time and participation. Subjects were divided randomly into two groups. One group experienced triggersets along with the visualizations provided by our system and the other only experienced the triggersets. This allowed us to examine how

people worked with the triggersets alone, and to compare the subjective experiences of this group with the visualization group.

ID	Gender	Occupation	Programming	Graphics
N1	F	Graphic Designer	No	CAD / Illustrator
N2	M	Postgrad Architecture	Yes	CAD
N3	M	Postgrad Computer Usability	Yes	No
N4	M	Postgrad Computer Usability	Yes	No
N5	F	Postgrad Astrophysics	Yes	No
V1	F	Architect	No	CAD
V2	M	Architect	No	CAD
V3	F	Postgrad Computer Networks	Yes	No
V4	M	Graphic Designer / Teacher	Yes	3DStudioMax
V5	F	Postgrad Philosophy / Writer	No	No
V6	M	Postgrad Computer Usability	Yes	No

Table 4: Details of the 11 subjects involved in the study. The subjects' gender, occupation, programming experience and graphics experience is shown.

5.4 Description of the study

In order to accomplish our aims, a study comprising two parts was devised. Each part involved the description of a VE. Visualization subjects were provided with a floorplan of the VE, timeline(s) for predictable sequences of events and a sequence diagram generated from the triggersets. The second group were only provided with the triggersets, a list of included objects, object details (e.g., position and orientation) and a textual description of the environment, including the locations (with coordinates).

For the first part of the study, a simple VE entitled `Bouncer Example' was conceptualised. The physical space of the Bouncer Example consisted of three locations, one of which was locked at the start of the VE. In addition to the User object, the VE contained a Bouncer, a Door, a Bell, a Suitcase and a Chalice. The Bouncer and Door had animations and the Bouncer and Bell had sounds attached to them. Various triggersets were already set up in the tool, which described the interactions that could happen. These were jumbled so that they were not in any kind of sequence. They all had meaningful descriptions. The aim of this part of the study was to see how well subjects could work out what might happen in the VE. The triggersets had been set up so that some could not execute without others having already been triggered, so that there was some sort of sequence. Attention was also paid to whether subjects noticed possible branches in the interactions depending on user actions. Table 5 displays the triggersets that were provided to the visualization group.

For the Debug part of the study, a VE entitled `Jail Example' was conceptualised. The space consisted of four locations. All of the locations were locked at the start of the VE (of course, the User begins in one of the locations). In addition to the User object, the VE contained a Jailer, a person called Sandra, three Doors, a Push-Button and a Chalice. As in the Sequence part of the study, various triggersets had been set up in the tool,

which described the interactions that could potentially happen. However, in this case, several mistakes had been introduced into the triggersets, which prevented some triggersets from going off or stopped the User from reaching the goal, which was to get into the location named Freedom (i.e., escape the Jail). The aim was for each subject to identify as many potential problems with the triggersets as possible. This was intended to test how well subjects could debug incorrectly programmed interactions.

Description	Trigger	Condition(s)	Action(s)
No Entry	User presses ‘x’	User is within 0.5m of Door AND Bouncer is within 1m of Door	Bouncer plays sound ‘No entry’ and Bouncer plays animation ‘Raise hand’
Entry Allowed	User presses ‘x’	User is within 1m of Door AND Bouncer is outside 1m of Door	Door plays animation ‘open’
World End	User presses ‘x’	User is within 0.5 m of Suitcase	World starts timeline ‘World end’
Bell Ring Effect	Bell sound ‘ring’ ends	Bouncer is within 1m of Door	Bouncer starts timeline ‘Bouncer move’
Bell Ring	User presses ‘x’	User is in Waiting Room	Bell plays sound ‘ring’
Close Door	Bouncer moves within 1m of Door		Door plays animation ‘close’

Table 5: The triggersets that were included in the Sequence part of the study. The columns show the triggerset descriptions, triggers, conditions and actions respectively.

The mistakes in the triggersets were created to try and cover typical mistakes that people make. We included examples which drew from all of the categories described in Section 5.2. The examples and the categories from which they draw are indicated in Table 6.

Mistake	Description	Error Category
A	Jailer does not move far enough and so proximity trigger is not executed	Spatial / Timing
B	No way specified to open a door behind which is a button to open the door to Freedom location, which is therefore never accessed	Implicit Assumption / Spatial
C	*Caught* and *Locked In* triggersets both execute and conflict	Sequencing & Logical
D	*Sandra Yes* and *Sandra No* triggersets both execute and conflict	Sequencing & Logical

Table 6: Interaction Programming Mistakes Introduced into the Visualization Study

5.5 Procedure

Each subject experienced the study individually and was then interviewed. Before the study began, the subject was given an introduction, in which the uncertain nature of interactions in VR was described. The triggerset interface provided by the tool for entering and describing interactions was also described in some detail, and the subject was introduced to the tool and how it worked. Following this, the instructions and VE descriptions for the Sequence part of the study were given to the subject. Both visualization and non-visualization subjects were given this basic description. Subjects who were to receive the visualizations were given a brief description of the provided diagrams, specifying that they were provided to help with the design and conceptualisation of VE interactions. Then subjects were given 20 minutes to examine the triggersets and object descriptions (and the visualizations for visualization subjects). They were allowed to make any notes or diagrams that they wished to help with their thinking. After 20 minutes they were asked to write down what might happen in their own words, indicating dependencies in how triggersets might execute and any other relevant details.

Thereafter, the instructions and VE descriptions for the Debug part of the study were given to each subject. A short paragraph described what was supposed to happen in the VE. It was then indicated that there were some problems with the triggersets which would stop them achieving the goals of the designer, or prevent triggers from being activated. Subjects were asked to specify any mistakes that they found and were also given 20 minutes to examine the triggersets for this part. The experimenter was present during the entire study to observe how the subjects worked with the tool and to answer any questions where subjects were uncertain about the working of the tool.

When subjects had finished with both parts of the study, a structured interview was conducted. The interview began with an discussion of the subject's written output from the first two parts of the study, where subjects were asked to take the interviewer through their answers and explain their process. Then all of the subjects were asked about the cognitive effort of working out the sequences of events and the interaction errors. They were also asked how they found working with the triggersets and how these might be improved. Visualization subjects were then asked more questions about the visualizations: how useful they found them, which they worked with better and found more useful, the specific tasks for which each visualization was most useful, how the visualizations interacted and how the subject might improve the process of working with them.

5.6 Analysis

A code was developed for scoring the accuracy of the sequence descriptions of the Sequence part of the study. Points were given for correct sequencing, awareness of branches, awareness of locations of objects and how these might change, and awareness of pre-conditions on actions. Table 7 displays the code that was used. A certain amount of redundancy was introduced in the coding, so that allowance could be made for subjects leaving out information that they did in fact know. For example, for the *Bell Ring Effect* triggerset (BE1 to BE4 in Table 7), subjects were given up to four points for

mentioning various bits of information, corresponding to the trigger, conditions and actions involved in the triggerset.

For the Debug part of the study, subjects were marked directly on how many errors they discovered. These codes yielded quantitative data, on which descriptive statistics could be calculated, such as the means for different groups. For the quantitative analysis, an independent marker examined the scripts using the code, in order to ensure that the marking was unbiased. In addition, content analysis [54] was performed on the transcribed texts of the interviews, to gain qualitative information about how subjects worked with visualizations and triggersets, and the cognitive effort involved.

Code	Description	Score
GS	Correct General Sequence	2
NE	No Entry Trigger	1
BR1	Bell Ring Trigger - Waiting Room	1
BR2	Bell Ring Trigger - Press x to ring bell	1
BE1	Bell Ring Effect - If Bouncer at Door when Bell Stops Ringing / Rings	1
BE2	Bell Ring Effect - Bouncer moves away from Door	1
BE3	Bell Ring Effect - Bouncer moves to Bell (and scratches head)	1
BE4	Bell Ring Effect - Bouncer moves back to Door	1
EA1	Entry Allowed - User goes to Door and press x to Open Door	1
EA2	Entry Allowed - Only when Bouncer not at the Door	1
WE1	World End - Inner Chamber	1
WE2	World End - Only Get to Inner Chamber if Door Open / Bouncer away	1
WE3	World End - In chamber User press x near Suitcase for end condition	1
WE4	World End - If press x near Suitcase, sound, suitcase invisible, chalice visible	1
CD1	Close Door - When Bouncer returns, closes Door to access Inner Chamber	1
CD2	Close Door - If User inside, then trapped when Door closed	1
Total		17

Table 7: The Code used to score the Sequence part of the visualization study. The code numbers refer to particular triggersets.

5.7 Results and Discussion

Because of the small size of the sample and in line with ethnographic guidelines [ref], only descriptive statistics were calculated from the quantitative data. These statistics and trends in the data show some interesting details.

5.7.1 Sequence

Visualization subjects got 72.5% of the sequence correct on average, while non-visualization people got 56.4% of the sequence correct. Table X indicates the scores of the subjects for the sequencing part of the study, and their totals.

ID	GS	NE	BR 1	BR 2	BE 1	BE 2	BE 3	BE 4	EA 1	EA 2	WE 1	WE 2	WE 3	WE 4	CD 1	CD 2	Tot
N1		1	1	1	1				1	1			1	1			8
N2	1	1	1	1	1				1				1	1	1		9
N3	1	1	1	1		1	1		1	1			1		1		10
N4	1		1	1	1			1					1				6
N5	2	1	1	1		1	1	1	1	1	1	1	1	1	1		15
V1	2		1	1	1	1	1		1	1	1	1	1				12
V2	2		1	1	1	1		1	1	1	1	1	1			1	13
V3	2	1	1	1	1	1			1	1		1	1		1		12
V4	2	1	1	1	1	1		1	1	1		1	1		1	1	14
V5	1		1		1	1	1		1		1		1	1	1		10
V6	2	1	1	1		1	1	1	1	1	1	1	1				13

Table 8: Scores of Subjects for Sequence part of Study, according to each part of the coding scheme. See Table 7 for a description of the coding scheme.

When we examine Table X, several interesting results emerge. Two points were given for a completely correct overall sequence (GS). Of the visualization group, only one subject did not gain full points, compared to all but one of the non-visualization group. This subject (N5) performed better on the Sequence part of the study than anyone else. This fact is interesting because of her background and the way that she performed the task. She has extensive mathematical education and experience. This allowed her to understand the coordinate system and the general mathematics of the interactions very well, e.g., how translations and rotations change the positioning of the objects. In addition, she drew an accurate floorplan and timed sequence diagram of her own, which she said enabled her to understand how the triggersets worked together. Therefore, this subject effectively re-created the visualizations that were given to the visualization group in order to understand the triggersets. If N5's score is taken as an outlier, the average of the non-visualization group drops to 48.5% and the difference between the groups increases from 15.1% to 24%. All of the non-visualization group did attempt to diagrammatically represent the triggersets; most used limited flowcharts. However, these were often inaccurate and so led them to incorrect conclusions. One subject (N2) claimed that if he had an hour and some graph paper, he would have drawn a floorplan, which would have made things much clearer. This highlights the fact that many people do not have the skills to distil complete and accurate overviews or visualizations, which enable a continuous picture of the discrete interactions.

The visualization group were much more likely to leave out details that were not part of a meaningful sequence, indicating that they were more aware of how triggersets fitted into an overall picture. The non-visualization group tended to write down everything that they noticed and not in any particular sequencing order. This suggests that they were much less aware of the triggersets as part of an overall sequence. For instance, if the User tried to approach the door to the Inner Chamber while the Bouncer was within a certain distance of it, the Bouncer would say 'No Entry' (NE Triggerset). This triggerset did not explicitly fit into any sequence, as it did not change the state of the VE. Only half of the visualization group noted this triggerset, whereas all but one of the non-visualization group referred to it. Similar results can be seen for CD1 and WE3.

The way that conditions provide constraints on multiple triggers executing (i.e., the hidden complexity of sequencing triggersets) was also noticed much more effectively by the visualization group. For example, the *World End* triggerset involves the User selecting a Suitcase object. It can only be executed from the Inner Chamber location (WE1) and the User can only go into the Inner Chamber when the Bouncer is not at the Door (WE2). These constraints were not explicitly stated in the *World End* triggerset. The first was a function of the location of the Suitcase and the second was a condition on the *Entry Allowed* triggerset, which was a dependency of *World End*. Only one of the non-visualization group (N5) noticed these facts, compared to four and five of the visualization group respectively. EA2 also gives a point for noticing that access is only allowed to the Inner Chamber when the Bouncer is away. Interestingly, because the constraint was explicitly stated in this case (as a condition in the triggerset), three of the non-visualization group noticed the point here.

Subjects in general seemed to have difficulty noticing consequences of executing triggers when they were implicit. For example, the triggersets specified that the Bouncer would move some distance away from a Door to the Inner Chamber, and then move back again. This door could only be opened when the Bouncer was not there. If it was open when he returned, he would shut it. The only way for the User to move the Bouncer away from the Door was to go to a different room and ring a Bell. If the User happened to be in the Inner Chamber when the Bouncer returned, the User would be locked in that room, with no way to escape, thus providing an end condition (or at least deadlock) for the VE. This possibility was not explicitly stated, as it relied on the movements of the User, which could not be pre-determined (CD2). Only two of the subjects, both from the visualization group, noticed this possibility.

In general, the visualization group did better in working out what was happening when translations and rotations were involved in the triggersets, perhaps because these actions were grouped on a timeline, so they could be viewed as one sequence. For example, in one triggerset, the Bouncer rotates and moves a set distance forward (BE2). All of the visualization group and only two of the non-visualization group noted that he was moving away from the Door. However, where more specific details about the Bouncer's movement were required, neither group did particularly well. The *Bell Ring Effect* triggerset involved combinations of translations and rotations which moved the Bouncer to a specified location. Subjects could tell that he moved (BE2), but were much less likely to know where he moved to (BE3 and BE4).

Obviously, because of the locations on the floorplan, the visualization group were much better at noting the locations of objects (BR1, WE1, EA2). The non-visualization group was only clear on these locations when they were explicitly stated in the triggersets.

5.7.2 Debug

Table 9 displays the results for the second part of the study. The letters, A to D, refer to the mistakes that were introduced. See Table 6 for a description of these mistakes. The percentages indicate what proportion of each group noted each error. In Table 10, the mistakes are shown divided into the types of errors which they involved and the percentages indicate what proportion of each group noted that type of error.

	A	B	C	D	TOTAL
Visualization	**17%**	**33%**	**17%**	**83%**	**37.5%**
No Visualization	**0%**	**20%**	**0%**	**40%**	**15%**

Table 9: Percentage of each group that detected errors. Error descriptions are displayed in Table 3.

	SEQUENCE / LOGICAL	IMPLICIT ASSUMPTION	SPATIAL	TIMING
Visualization	**50%**	**33%**	**25%**	**17%**
No Visualization	**20%**	**20%**	**10%**	**0%**

Table 10: Percentage of each group that detected different types of errors.

As can be seen from the tables, the visualization group detected more than twice as many errors as the non-visualization group. Overall, the percentage of errors noted was still not good. However subjects were limited to 20 minutes to familiarise themselves with the VE and find errors, which may have influenced the results.

If we examine the errors that were detected broken into the various error types, some interesting results appear. The visualizations seem most helpful for sequencing errors, probably because the sequence diagram indicates a probable sequence. Without this, the triggersets must be manually connected to each other, based on the result of each one being executed. Timing errors are almost impossible to find without some way to view the final product or step through the events. However, even the limited help provided by the sequence and floorplan enabled the visualization group to detect errors to some extent.

5.7.3 General Results and Content Analysis

There were no noteworthy differences between those with programming experience and those without. The same is true of gender. Below, we examine the results against our original aims.

In terms of testing the user interface of the triggersets, the study showed that subjects were in general comfortable working with the interface. All of the subjects from both groups found the triggerset formalism easy to work with. They understood how the trigger-condition-action triads worked and when they might be executed. It was when the triggersets had to be ordered in a sequence that non-visualization subjects began to have difficulties. Only non-visualization subjects mentioned problems with thinking linearly. “When you read something, you say OK so he goes there but you don't think that there might be something stopping movement - very hard to think about how triggersets all fit together - got more used to realising that anything can happen at any moment, but the order of things was still hard to realise.”

In terms of observing how people worked with the visualizations, the study had very positive results. Subjects all worked in different ways with the visualizations and

triggersets. This justifies the flexibility of the tool in allowing for designers having different work processes. The visualization group all looked at multiple visualizations to try and work out what happened. Only one subject mentioned having problems with constantly switching between visualizations. All of the visualization group found the visualizations clear and useful but found each one useful for different things, e.g. 'I had all three (visualizations) open at the same time. Then, if you don't understand the sequence you can look at the timeline.' And 'The triggersets are basically just the details, just the details of the rest (the visualizations)'. They all felt that the tasks would have been much harder without the visualizations. A few mentioned that the limited interactivity provided was really helpful in terms of working with each diagram.

They found the floorplan most useful: to orient, give a concrete sense of the space and where the objects are in relation to each other "Once you coordinate between the physical locations, you can see how you need to move." In fact, half of the visualization group stated that they could not have reconstructed the sequence without the floorplan. Timelines were useful for noticing a predictable sequence "Used the timeline for Bouncer Move to see how he went away." One subject, who did not use the timelines much stated, "I did not use timelines much to examine interactions because they were simple, but they would be very useful to make actions - work very nicely. For design, I like the timeline. It is important as both a visualization and a construction tool." Mistakes in the Debug part of the study were made more obvious by the sequence diagram: "The sequence diagram is useful for seeing how the triggersets relate, their order and what activates what." "Could walk anywhere, but the VE only reacts like the sequence." Even those who found the sequence diagram less useful stated that they were useful to "check up after your own analysis of the triggers". Subjects wanted more obvious interactivity from the sequence diagram. However, as they were unable to enter interactions, they could not see how the sequence changes as the designer changes the triggersets. Therefore they could not experience this form of interactivity.

In terms of finding out whether people who experienced visualizations would perform better in sequencing and debugging interactions, the study also had very positive results. Most of these have been indicated above, in Sections 5.7.1 and 5.7.2. The visualization group understood the sequences much more accurately than the non-visualization group. They also found errors in the triggersets much more successfully. In addition, all but one of the non-visualization group stated that visualizations would have made their task much easier. Three specifically mentioned some kind of flowchart and a floorplan and one mentioned organising the triggersets in a timeline.

5.7.4 Implications for design

Our study also indicated areas where we could improve the system. Subjects underused the timelines, as they were connected to objects and so more difficult to access. This means that the visualizations must be made more accessible. We are considering listing the timelines together with their triggers as another method of access. People need to be able to access each visualization from the other visualizations, so that they do not forget about them and to make it easier to cross-reference. Some visualization of timeline activity on the floorplan would help people to work out sequences.

Subjects did not often drill down into triggersets and object details to find out more, e.g., the exact angle of rotation or the location of an object. For the visualization group, the diagrams indicated many of these details, but other details must be made more obvious and easily accessible. It would be helpful to group triggersets according to various criteria, such as locations in which they might execute and objects which are involved. Subjects indicated that providing this context would help them to understand how triggersets work together.

People need more help with working out the consequences of rotations and translations, in terms of where an object will end up after a sequence. These could also be played out on the timeline.

6. Conclusions

We have described a system for helping content experts to design, implement and debug interactions in a VE. We designed the system carefully, using previous research and appropriate theories to support our decisions. The system combines simple event-action input with visualizations to scaffold the design and implementation process for the novice user.

We also conducted a study to assess the effectiveness of parts of the system. This study showed that those who received visualizations performed better than those who did not, both in terms of detecting sequences in triggersets and in terms of debugging errors. These are very positive findings for the usefulness of visualizations in aiding design decisions for VE interaction design.

In terms of the specific aims of the study, the results were also positive and informative. We wanted to test our user interface and the triggerset mechanism. We found that all subjects, even those who did not receive visualizations, were positive about the triggersets and found them easy to understand. Sequencing them introduced problems, but these were far fewer in the visualization group. Introducing various ways of organising the triggersets should also help with sequencing difficulties.

We also wanted to observe how people worked with the visualizations both in terms of understanding them and utilising multiple windows. We found that subjects understood the visualizations and worked well with them. The interactivity that was added was also very positively received.

These results show that the idea of using visualizations to understand and debug VE interactions is a highly successful one. Additionally, people use multiple interacting visualizations effectively. We also learnt a lot about how people work with visualizations and what further aid should be given to support them. These lessons will be used in the next iteration of system design.

6. References

[1] H.G. Hoffman, A. Garcia-Palacios, A. Carlin, T.A. Furness III, C. Botella-Arbona, Interfaces that heal: coupling real and virtual objects to treat spider phobia, in: International Journal of Human-Computer Interaction, 16(2), 2003, 283-300.
[2] I. Ladeira, E. Blake, Virtual san storytelling for children: Content vs. experience, The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage, VAST 2004, Brussels, Belgium, The Eurographics Association, 223-231.
[3] D.A. Bowman, E. Kruijff, J.J. LaViola, I. Poupyrev, An introduction to 3-D user interface design, in: Presence, 10(1), February 2001, MIT Press, 96-108.
[4] M.J. Schuemie, P. van der Straaten, M. Krijn, C. van der Mast, Research on presence in virtual reality: a survey, in: Journal of CyberPsychology & Behaviour, 4(2), 2001, 183-201.
[5] D.Zeltzer, Autonomy, interaction and presence, in: Presence: Teleoperators and Virtual Environments, 1(1), 1992, 127-132.
[6] K. Hinckley, R. Pausch, J.C. Goble, N.F. Kassell, A survey of design issues in spatial input, in: Proceedings of UIST '94, ACM Press, California, 213-222.
[7] K. Harada, E. Tanaka, R. Ogawa, Y. Hara, Anecdote: a multimedia storyboarding system with seamless authoring support, in: ACM Multimedia (1996), 341–351.
[8] T. Schiphorst, T. Calvert, C. Lee, C. Welman, S. Gaudet, Tools for interaction with the creative process of composition, In: CHI '90 Proceedings, April 1990, ACM Press.
[9] O. Balet, P. Kafno, F. Jordan, T. Polichroniadis, The VISIONS project, in: O.Balet, G.Subsol, P.Torguet (Eds.), Virtual storytelling: using virtual reality technologies for storytelling (Proceedings of ICVS 2001), Springer-Verlag, Berlin, 2002, 90–99.
[10] Virtools The Behaviour Company, http://www.virtools.com.
[11] D. Sanchez-Crespo, Product review: Virtools dev 2.0. In: Gamasutra, April 2002, http://www.gamasutra.com/features/20020809/crespo 01.htm.
[12] A. Coleburne, T. Rodden, K. Palfreyman, VR-MOG: A toolkit for building shared virtual worlds, in: M.Slater (Ed.), Proceedings of FIVE (Framework for immersive virtual environments) Working Group Conference (1995) 109–122.
[13] M. Conway, S. Audia, T. Burnette, D. Cosgrove, K. Christiansen, R. Deline, J. Durbin, R. Gossweiler, S. Koga, C. Long, B. Mallory, S. Miale, K. Monkaitis, J. Patten, J. Pierce, J. Shochet, D. Staack, B. Stearns, R. Stoakley, C. Sturgill, J. Viega, J. White, G. Williams, R. Pausch, Alice: lessons learned from building a 3d system for novices, in: CHI Proceedings, 2000.
[14] R. Pausch, T. Burnette, A.C. Capehart, M. Conway, D. Cosgrove, R. DeLine, J. Durbin, R. Gossweiler, S. Koga, J. White, A brief architectural overview of Alice, a rapid prototyping system for virtual reality, in: IEEE Computer Graphics and Applications, 15(3), 1995.
[15] AgentSheets Reference Manual, http://agentsheets.com/.
[16] A. Repenning, T. Sumner, Agentsheets: A medium for creating domain-oriented visual languages, IEEE Computer 28(3), 17-25.
[17] Sim City 3000 Strategic construction game user manual, http://simcity.ea.com/.
[18] A.J. Ko, B.A. Myers, A framework and methodology for studying the causes of software errors in programming systems, in: Journal of Visual Languages and Computing, 16, 2005, 41-84.

[19] A.J. Ko, B. A. Myers, H.H. Aung, Six learning barriers in end-user programming systems, in: IEEE Symposium of Visual Languages and Human Centric Computing, Rome, 2004, 199-206.
[20] A.F. Blackwell, First steps in programming: a rationale for attention investment models, in: Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments, Sept 3-6 2002, Virginia, USA, 2-10.
[21] C. T. Fosnot, Constructivism: a psychological theory of learning, in: Constructivism: Theory, Perspectives and Practice, Teachers College Press, 1996, (chapter 2).
[22] E. von Glasersfeld, Introduction: aspects of constructivism, in: Constructivism: Theory, Perspectives and Practice, Teachers College Press, 1996, (chapter 1).
[23] C. Winterbottom, E. Blake, Designing a VR interaction authoring tool using constructivist practices, in: 3rd International Conference on Virtual Reality, Computer Graphics, Visualization and Interaction in Africa, ACM Press, November 2004.
[24] M. Petre, A.F. Blackwell, T.R. Green, Cognitive questions in software engineering, in: J. Stasko, J. Domingue, M. Brown and B. Price (Eds.), Software Visualization: Programming as a multi-media experience, MIT Press, 1998, 453-480.
[25] M. Baldonado, A. Woodruff, A. Kuchinsky, Guidelines for using multiple views in information visualization, in: Proceedings of AVI 2000, Italy, ACM Press.
[26] Z. Hendricks, G. Marsden, E. Blake, A meta-authoring tool for specifying interactions in virtual reality environments, in: Proceedings of Afrigraph 2nd International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction in Africa, 2003.
[27] D. Norman, The Psychology of Everyday Things, Harper Collins Publishers, USA, 1988.
[28] B. Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Co., Massachusetts, 2nd edition, 1992.
[29] S.K. Card, J.D. Mackinlay, B. Shneiderman (Eds.), Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann Publishers, California, 1999.
[30] N.C. Shu, Visual Programming, Van Nostrand Reinhold Company, New York, 1988.
[31] P. Romero, R. Cox, B. du Boulay, R. Lutz. A survey of external representations employed in object-oriented programming environments, in: Journal of Visual Languages and Computing, 14(5) (2003), 387-419.
[32] G-C. Roman, K.C. Cox, Program visualization: the art of mapping programs to pictures, In: Proceedings of the 14th International Conference on Software Engineering, May 1992, 412-420.
[33] B.A. Myers, Taxonomies of visual programming and program visualization, in: Journal of Visual Languages and Computing 1, 1990, 97-123.
[34] J.F. Pane, A programming system for children that is designed for usability, PhD Thesis, Carnegie Mellon University, Computer Science Department, CMU-CS-02-127, Pittsburgh, May 3, 2002.
[35] B.A. Myers, J.F. Pane, A. Ko, Natural programming languages and environments, in: Communications of the ACM, 47(9) (2004), 47-52.
[36] A.J. Ko, B. A. Myers, Designing the Whyline: A debugging interface for asking questions about program behaviour, in: CHI 2004, Vienna, Austria, 151-158.
[37] M. Petre, A.F Blackwell, Mental imagery in program design and visual programming, in: International Journal of Human-Computer Studies 51 (1999), 7-30.

[38] M. Kavakli, M. Suwa, J. Gero, T. Purcell, Sketching interpretation in novice and expert designers, in Gero, J.S., Tversky, B (Eds.) Visual and Spatial Reasoning in Design, Key Centre of Design Computing and Cognition, University of Sydney, Australia, 1999, 209-220.
[39] E. Do., M.D. Gross, Drawing as a means to design reasoning, in Visual Representation, Reasoning and Interactions in Design Workshop notes, Artificial Intelligence in Design '96, Stanford University.
[40] A.F. Blackwell, Diagrams about thoughts about thoughts about diagrams, in: M. Anderson (Ed.), Reasoning with Diagrammatic Representations II: Papers from the AAAI 1997 Fall Symposium, 77–84.
[41] C. Eastman, New directions in design cognition: studies of representation and recall, in: C. Eastman, M. McCracken, W. Newsletter (Eds), Design Knowing and Learning: Cognition in Design Education, Elsevier Science, Amsterdam, 2000.
[42] A.F. Blackwell, K.N. Whitley, J. Good, M. Petre, Cognitive factors in programming with diagrams, Artificial Intelligence Review 15(1) (2001) 95-113.
[43] K. Kaur, Designing virtual environments for usability, Phd thesis, Centre for HCI Design, City University, London, June 1998.
[44] C. Fencott., Towards a design methodology for virtual environments, in: Workshop on User Centred Design and Implementation of Virtual Environments, York, England, 1999.
[45] C. Beirowski, H. Vermuelen, Experiences with virtual reality accessibility in an African context, in: Workshop proceedings of the IEEE VR 2004 Conference, 14-18.
[46] J. Tangkuampien, Virtual Environment Authoring Interface for Content-Expert Authors, Masters thesis, Computer Science Department, University of Cape Town, February 2005.
[47] G. Zachmann, A language for describing behaviour of and interaction with virtual worlds, in: Proceedings of ACM Virtual Reality Software and Technology(VRST) Conference, July 1996.
[48] E.R. Tufte, The Visual Display of Quantitative Information, Graphics Press, Cheshire, 1983.
[49] C. Fencott, Virtual storytelling as narrative potential: towards an ecology of narrative, in: O.Balet, G.Subsol and P.Torguet (Eds.), Virtual Storytelling: Using Virtual Reality Technologies for Storytelling (Proceedings of ICVS 2001), 90–99.. Springer- Verlag, Berlin, 2002.
[50] D. Harel, On visual formalisms, in: Communications of the ACM, 31(5) (1988), 514–530.
[51] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, M. Trakhtenbrot, STATEMATE: a working environment for the development of complex reactive systems, in: IEEE Transactions on Software Development, 16(4) (1990), 403–414.
[52] D. Wolber, A multiple timeline editor for developing multi-threaded animated interfaces, in: ACM Symposium on User Interface Software and Technology, 1998, 117–118.
[53] Spradley, Participant Observation. Holt, Rinehart and Winston. New York, 1998.
[54] P. Banister, E. Burman, I. Parker, M. Taylor, C. Tindall, Qualitative Methods in Psychology: A Research Guide, Open University Press, Buckingham, 1994.