Constructivism, virtual reality and tools to support design

Cara Winterbottom
Collaborative Visual Computing Lab
Department of Computer Science
University of Cape Town
+27 21 650 2663
cwinterb@cs.uct.ac.za

Edwin Blake
Collaborative Visual Computing Lab
Department of Computer Science
University of Cape Town
+27 21 650 2663
edwin@cs.uct.ac.za

ABSTRACT
This paper describes a process for creating a design tool, which is based in constructivism. The process is described for the creation of a tool to help novices in designing virtual environment interactions, however it can be generalized to other design domains. The process consists of four steps: first constructivist values of atomic simplicity, multiplicity, exploration, control and reflection are distilled. Next, expert practices are researched and reframed in terms of the constructivist values. Thirdly, novice processes are examined and understood in constructivist terms. Lastly, prototypes are created and shown to target users. These steps are iterated until the designed tool is satisfactory.

Categories and Subject Descriptors
H.5.2 [Information interfaces and presentation]: User interfaces – prototyping, theories and methods; H.5.1 [Information interfaces and presentation]: Multimedia information systems – artificial, augmented and virtual realities, evaluation/methodology

General Terms
Design, Human Factors

Keywords
Design, Virtual Reality, Constructivism

1. INTRODUCTION
It is generally recognised that design is difficult. It is made even more intractable by the fact that it can be applied across domains to yield vastly different design artefacts with vastly different requirements. Any account of a design process is necessarily situated within the domain for which the design is being carried out. If one examines accounts of expert designers across disciplines, however, it is possible to find over-arching principles and practices of good design. We present a design process for a virtual reality (VR) design tool for novices that is based on constructivism. We argue that constructivism opens up many possibilities for how design could be effectively accomplished and learned, and for how the interfaces of a design tool can help novices to create valuable design artefacts while learning. We also argue that this process and the underlying theory have more generic applicability than only the specific domain.

DIS 2008, February 25-27, 2008, Cape Town, South Africa.

As technologies become more wide-spread, they become more accessible. End-users, who do not necessarily have the training or much experience with technology, are taking a greater part in customising their software and creating their own products, for example, presentations, video and sound editing. Within the arena of 3D software, 3D gamers are more and more able to modify their favourite games through scripting and changing 3D models [35]. Virtual reality (VR) is closely related to 3D games, and end-user facilities for creating virtual environments (VEs) are also being developed [44]. We want to make VR design more accessible, so that end-users can creatively design and develop their own VEs in the field in which they are experts (e.g., marketing materials, architecture walkthroughs, or cultural exhibits). Even if this is only accomplished in an inexpert way, the results will still enable better communication between groups in a project team. For example, the programmers will have a much better idea of what the content expert wants with a dynamic prototype. This potential for improved communication within design teams is valuable not just within the field of VR and games, but in all design domains. Early development of core design principles and successful communication of these should improve the efficiency and effectiveness of any design cycle.

In this paper, we describe the process of creating a tool for designing and implementing the interactions within VEs. We have not seen constructivism applied to interactive system design previously. There have been many accounts of constructivism applied to creation of educational content for interactive teaching systems [16, 45]. However, here we apply it *both* to the content of our tool (which is to help in the design of VE interfaces) *and* to the design of the interface to our tool. First, we lay out the steps of our process. Then we discuss in detail each step with reference to our specific domain and for more generic design, with background information and reference to studies conducted. Finally, we examine the implications of our process in the resulting system and for more general design products.

2. BEGINNING
"*The most brilliant concept is useless unless you can think of the way that people will play the game. In the end, game design comes down to interface design—the key to making games playable is how you'll get people to interact with your concept and how simple the interface is.*" (Peter Molyneux, designer of games such as *Theme Park* and *Black & White*, [35] page 77).

The original brief of this project was to research interfaces for designing interactions in a VE, and to make this work accessible to non-programmers. This expanded to some extent to become helping novice designers to conceptualise and

visualise the interactions in a VE, focussing on design aids that disentangle the complexity. Designing interactions within a VE does not on the face of it appear to be difficult, and it is certainly not difficult to design simple interactions. The problem here lies in designing an effective final VE. In VR, any narrative or goal has to be flexible enough to be found by the user, without their being forced along a particular path. Of course, you could force the user along a path and our design tool makes allowances for this to happen through scripted sequences and control of the user camera (this is the representation of the user in the VE, sometimes along with an avatar); but the innovation of VR is that the user can determine the progression of events. This, above all, is what distinguishes it from films. Because the events of a VE are dependent on the actions of the user(s), they cannot be completely predetermined. To a large extent the static environment can guide the user: one can create perceptual opportunities [15]. However, dynamic events are even more effective, and also engage the user in the narrative of the VE. Apart from other actors, who can interact with the user in different ways depending on time, place and sequence, the environment can also react to the user, e.g. from the obviousness of signs appearing to changes of lighting [8].

As well as designing interactions, the VE creator must implement the design. This has traditionally only been done through programming, although there now exists software, such as Virtools, which allows users to implement graphically [44]. In order for the designer to be aware of what is technically possible and/or feasible in a VE, an effective design tool should allow the designer to implement the design (or at least make technological constraints apparent).

Because of our requirements (interfaces that work for non-programmers), we looked outside computer science for ideas or practices that could inform our research. In psychology and educational theory we found constructivism [16, 37, 45]. Constructivism states that knowledge is constructed by individuals through their interactions with others and their physical environments. In constructivism there is no abstract truth, but we can construct knowledge by creating a coherent network of viable explanations. It focuses on learning through doing. As such, the theory supports notions of multiplicity of representation and explanation, reflection on actions and their consequences, support for exploration, help in the form of scaffolding that can fall away rather than marking out an exact path, and user control of the process.

Practical application of constructivism has traditionally been constrained to explicitly educational products. For example, there are many accounts in the literature about using constructivism in hypertext systems [e.g. 42, 29]. Hypertext is a technology that lends itself to the application of constructivist principles, as the hypertext user actively directs her reading and defines her own path through the information, which gives control over the knowledge-building experience. The World Wide Web is a very large hypertext system and so the ideas expressed above apply here too. A lot of learning material on the Web is organised so that students can learn in a self-directed and active way. One could argue that any exercise in finding information on the Web is implicitly using constructivist learning principles [46]. Our approach is related to this idea, and based on the concept that we learn new things all the time in a constructivist fashion. And so learning interface design can also be based on constructivist approaches. In addition, the user who has to learn a new interface is also busy with a knowledge building task.

2.1 Restrictions

There are three main parts to designing and developing a specific VE: deciding on the hardware and input mechanisms, creating the content and creating the interactions. Since we were focussing on interactions within the VE, we decided to use traditional, low-cost hardware (desktop, keyboard and mouse) so as not to confound the study results by obtaining results about interactions with the hardware. In addition, the target group is content experts, who do not necessarily have graphical design experience in their own fields and probably do not know much about the technology of VR. We wanted to use hardware with which they were most likely to be familiar in their own fields. For similar reasons, we did not investigate creation of content for VR. This is a very different problem to that of designing interactions.

Another restriction on the design tool was that we did not want to introduce any artificial intelligence into the actors. More and more, in VE applications, actors are provided with high-level behaviours which guide their actions. We decided not to include such a system so as not to confound the research ideals. We wanted to start with the basics and work out how to help users at this level. Programming behaviours is complex and requires an AI engine to be introduced.

3. CONSTRUCTIVIST DESIGN METHOD

Constructivism informs our process for designing and developing an interactive system. We focus on making it highly usable and friendly for naïve users. Design and the problem domain, VR, are both difficult subjects. So we aimed to simplify everything else.

The theory was applied at two distinct levels during the design process. The tool had to be useful to novice designers of VE interactions. Therefore, there needed to be support for learning about VE interaction and its facets. Since constructivism is a theory of knowledge and how it can be constructed and managed, it seemed appropriate to use it to assist the users of the design tool in their own design processes. The process was as follows:

1. Distil practical values from constructivism to guide design.
2. Examine processes of expert designers and experts in the domain and rephrase them in terms of the design values.
3. Investigate the practices of novices in the domain area (VE design) in order to understand their specific problems.
4. Iteration: create a prototype and show it to the target audience. Let them use it. Focus on design values and their effects in questions afterwards. Repeat steps 2 to 4 until goal is met.

3.1 Practical Values of Constructivism

In order to use constructivism as a base for the development of a design tool we had to extract usable principles from it. Constructivism is like the classical Hydra: it has many different heads, often depending on the field to which it is applied, but the heads all connect to the same body of thought, with some minor variations. For example, in psychology the focus is usually on deconstructing existing practices or theories so that the parts which are constructed become more obvious [45]. In education, as stated by theorists such as Donald Schön, the focus is on how to make learning more effective by helping learners to construct knowledge themselves [37,16]. All the variations and the different ways in which constructivism is framed (it even has slightly different names: constructionism is

also used), make it difficult to uncover the core values of the theory. Through careful comparison of different constructivist accounts, we distilled five distinct values with which we could work.

3.1.1 Atomic simplicity

This is the concept that complexity arises from the ways in which simple atomic parts are connected together. So, the smaller parts of a thing can be specified very simply, and linked together in complex ways. This is the guiding principle of hypermedia systems, such as wikis. In creating interactions, this suggests that the novice designer should be able to specify individual interactions very simply and piece them together to yield a rich and interesting dynamic experience. The context in which interactions take place provides variation and richness.

3.1.2 Multiplicity

The idea of a single, correct truth does not exist within constructivism. From this comes the concept that there are many ways to an end, and not one single process. Just as we can construct our knowledge and world in myriad ways, so we can examine and analyse them from multiple perspectives and using multiple methods. So the value of multiplicity is that complexity can be broken down and understood by focussing on different aspects of it. A design tool can help the novice designer to see and interpret these multiple perspectives.

One way of doing this is to provide multiple representations, covering different aspects of a problem. However, the problem does need to be understood as a whole, which means that the representations must be conceptually related to each other. This can be done via techniques like juxtaposition, synching and embedded linking. For example, a 3D window can be viewed alongside a 2D floorplan of the area and a flowchart showing narrative of the world. This juxtaposition allows the designer to see that the representations all refer to the same problem space, but emphasize different aspects of it. Buttons and hyperlinks between all three views help the designer to juxtapose them. In addition, while interactions are happening in the 3D world, the other representations can be synched to this, showing the progression of action in all views. Thus, the interrelatedness of the representations can be indicated to the designer.

3.1.3 Practical Exploration

In constructivism, learning and constructing knowledge are active tasks. By making exploration attractive, a design tool can make it easier to learn both how to use the tool and how to design in the problem domain. One way in which this can be done practically is through the way in which errors are handled. If errors are treated as an integral part of the exploration process, they can be used to provide feedback about what is required to complete a design task. Users of the tool are able and allowed to make mistakes and view their consequences to provide concrete insight about design choices. This also suggests that errors must be relatively easily forgiven: any design tool should be robust enough to handle errors without breaking down.

3.1.4 Control

Allied to the concept of active construction of knowledge is the idea of personal control. People gain power over their learning processes by actively constructing their own knowledge. In education, the concept of scaffolding is described for providing help to learners. Here guidance is provided in the form of artefacts, advice and tutorials, which fall away when the learner has constructed the knowledge and skill to accomplish the task alone. The educator becomes a coach rather than a teacher [37]. A design tool, ideally, should provide dynamic help, which can be tailored to the user's capabilities. This is much harder than when an educator is available to assess the situation, as a software system is less able to judge the user's state. However, various levels of guidance can be provided, such as contextual help, assisted external representations of the process, pop up messages and comprehensive templates or tutorials based on domain knowledge.

3.1.5 Reflective Process

While activity is important in constructivism, the process of constructing knowledge requires acting with reflection so as to build effective connections between bits of knowledge, with an understanding of their consequences [37]. If a design tool can promote reflection about the design process and its results, it can be an effective aid to learning and understanding the problem domain. Multiple external representations of the design process can be used to promote reflection by enabling the designer to examine his or her work from different levels and angles [6, 11, 14]. Allowing for early prototyping and feedback will add to this power, by enabling contrasts with the representations. Support for iterative process is another way to foster reflection, by creating a space for the designer to reflect on the output of one iteration, and then apply the new knowledge to the next iteration.

3.2 Applying Constructivist Principles

While we were examining constructivism, an interesting and unforeseen side-effect became apparent. We had been considering constructivism from the point of view of the potential users of our tool: how the tool could be designed to minimise its learning curve and maximise its users' potential to create designs, specifically within the problem domain. As we worked with the values, we became aware that they were serving us in a completely different way: they were informing *our* design process. In other words, as well as using constructivism to create an interface for effective VE design, we were using the constructivist principles to guide us in *designing* an effective design tool.

Each part of our system was designed to be simple and modular, so that it could easily be plugged together and link up in highly interconnected and complex ways. Thus, if necessary, any part could be replaced by a part with slightly different functionality, as long as it connected to the whole in the same way. For example, we used the open source render engine, Irrlicht, to create our 3D view for prototyping the designed VR product. This can be replaced with another engine very easily. The mouse and keyboard can be replaced with a different system (such as a CAVE system, where images are projected onto surfaces around the user to create a 3D effect) that allows alternative input mechanisms.

To gain alternative perspectives on computer design tools and their effectiveness, we examined other tools which had been effective in completely different domains. For example, as well as VR design tools [44], we examined a choreography tool [38], a multimedia presentation tool [21] and a tool for creating simulations [33]. By examining design tools from diverse domains, we could find their similarities and the ways in which they were tied to their problem domains. They all provided multiple views of the design process at different levels of granularity; they all provided direct manipulation interfaces as this was more natural for their users; and they all provided immediate feedback.

While we know the field of VR, we were novices at using many of the systems which we explored (such as the choreography and simulation tools); therefore we engaged with these at the same level as new users of our own tool would engage with it. Therefore, we could experience the effects of each system and construct knowledge about design requirements for our tool.

3.3 Expert Processes

"*Creating a game usually follows a topdown methodology, with game vision on top of the pyramid, game ideas and features in the middle, and low-level game mechanics on the bottom. According to Goodman, many game designers concentrate too much on the top two levels and not enough on how the play mechanics will work in the game. "The bottom level is the hardest; the implementation of the details is key," says Goodman. He also says most design docs seem to rely on the first two levels, when the emphasis should be on the implementation of the ideas into the computer and not just the ideas themselves.*" ([35], quoting Rick Goodman, lead designer of *Age of Empires*, page 84).

Our next step was to examine general expert design practices, and the practices of experts in VR authoring, in the light of our constructivist values. Expert design practices in any field can provide general design guidelines. Design has been defined as an ill-structured problem space, with no predefined goals or constraints [12, 41]. This view means that creating a design is a problem-solving exercise. This is very difficult to do without constraints, especially for the novice [28]. Experts have constraints built in because of their experience and knowledge of how things work. A design-aid can help novices by providing the basic constraints and goals which they lack, with expert knowledge built into the tool and scaffolding informed by domain specific information. General expert design practices, like user interface design guidelines [30, 39], are useful as a starting point for dealing with common problems. For specific goals and constraints, we turn to the practices of domain experts.

We examined studies of expert design practices and found several guidelines. A study comparing architectural experts and novices in their sketching behaviour found that experts sketched more and included more alternative perspectives or designs [26]. In this way, experts work through many iterations and require the ability to play with their designs. They also revisit their designs and discover more in depth implications of the sketches [14, 36]. Expert programmers draw external representations less often, but state that they usually mentally visualise problems and ideas in graphical or other metaphorical ways [30]. Various expert designers questioned about ubiquitous computing design tools said they wanted multiple representations of the design problem, so that they could choose the appropriate view for each stage of the design. They also wanted simulation modes, knowledge support, and general support for iteration and prototyping. The ideal design tool also needed to have support for communicating designs throughout project teams [13]. These findings are echoed by other studies on the creativity of experts [10, 24], which stress the necessity of learning by doing and being able to reconceptualise and restructure a problem to allow insights to emerge.

We also examined studies of domain experts: that is, experts in the field of designing VEs. From these studies and our own extensive experience in VE design and implementation, we distilled more specific guidelines for creating VEs. Many points similar to those made by expert designers also emerged. Fencott [15] describes in high-level terms how to design a VE in order to guide VR users towards goals implicitly. He defines perceptual opportunities as creating narrative paths through a VE, via the use of attractors, connectors, retainers and surprises in the landscape. He advocates creating perceptual maps describing these opportunities so that the designer can focus on perceptual design issues. This strategy is, in effect, deconstructing the design process and focussing on parts of it with different representations. It also allows the designer to communicate his ideas to guide the user to other team members, via a common language.

Kaur [25] compiled general VE design guidelines for her PhD thesis. She then created a taxonomy of generic 3D usability guidelines. For example, the category *navigating* includes *keeping navigation pathways clear*. This was intended as a resource for VE designers to be able to actively test the effectiveness of their VEs by using a framework of 3D usability guidelines. This allowed designers to reflect on the effectiveness of their VEs by heeding simple guidelines in each interaction.

Bowman et al. [7] created a VR testbed where they explored the effectiveness of various travel, selection and manipulation techniques in immersive VR (participants wearing data-gloves and head-mounted displays). Here, the authors were learning about interaction techniques and their effectiveness by creating examples of interesting techniques and using them. In an iterative manner, they implemented interaction techniques and tested them in a variety of typical VR situations.

There is a lot of information about strategies of game designers [18, 35]. In interviews with successful game designers, Marc Saltzman [35] asked them about their design secrets. These designers work in large production teams, and clear communication is very important. Storyboards and detailed design documents are important for conveying the look and feel of a game. Equally important is the ability to prototype early so that designers can view their work 'in game', and understand the player point of view. For example, Brian Reynolds who has worked on games such as *Colonization* and *Alpha Centauri*, states:

"Strategy games are extremely complex to design—although the individual components look deceptively simple, having a lot of 'simple' moving parts makes for a very complex overall balancing task… To balance all the moving parts correctly, there's no substitute for actually playing your own game—the combinatorial explosion from all the moving parts makes it impossible to truly anticipate or tune results 'on paper' in a design document. The sooner you get your game running, the sooner you can actually get to work on making the game fun and making it balanced. Both fun and balance tend to be taken for granted by novice designers." ([35], page 72)

This advice is echoed by other designers in the same chapter. There is also a lot of exciting research being conducted into creation of various types of games, which address many of the issues with traditional game design (as the quotes in this section show). For example, work by Tracy Fullerton and the Game Innovation Lab at USC [17], where "play-centric design" is followed, and students create innovative games by focusing on player experience in iterative cycles and all levels of what makes games work. The problem of creating an effective innovative game is examined from all these perspectives. Other recent examples of innovative game design include a mixed-reality Location-Based-Game, Battleship, developed by Bidwell et al. at James Cook University [5] and Mermaids, a massively multiplayer online game (MMOG) developed by Pearce et al. from Georgia Tech's Emergent Game Group [31].

There is also much ongoing work on interesting design strategies in the MUD and MOO communities. Here, players, engineers and designers play and work together to create large persistent worlds with multiple narratives playing out [3, 34]. These narratives may interact to great meta-narratives or may continue in parallel lines. The complexity often emerges from the multiple levels of input from many committed participants.

The guidelines that we distilled from this research needed to be related to our constructivist values and incorporated into the design of our tool. The fact that experts often create external representations and revisit these throughout the design process ties in with values of multiplicity and reflection. Experts use their sketches as a reference for thinking and reasoning about their designs. Research on external representations suggests that they are useful in promoting reflection and a deeper understanding of the subject of the representations [6, 11, 14]. Novices may not know how to create useful representations or use them effectively for design. However, Eastman [14], in a survey of representations used in design, states that novices can learn from viewing and working with external representations, so that in time these are internalised and part of the designer's reasoning tools. Therefore, providing domain specific external representations of pertinent parts of a design can act as scaffolding for novices, most effectively if expert knowledge about how to read the representations is included (e.g., in the form of templates and context sensitive help).

Almost all experts surveyed called for early prototyping and the ability to communicate designs. If a VR tool provides the ability to prototype early in 3D, alongside the more simple external representations which can focus on specific aspects of the design (e.g., a floorplan or timeline), then it accomplishes this. The prototype and representations can be used to communicate design ideas effectively to other members of the design team. Exploration and control are fostered by this, as designers gain immediate feedback about their ideas in 3D. They can interact with this and the juxtaposed alternative representations to play with their designs (without having to pass the work onto a programmer to see it realized). If 3D content is not yet available, limited prototyping can be done with the simpler representations.

Lastly, many experts talked about putting together the components of a design into an effective, more complex whole. This dovetails with the constructivist idea of keeping simple atomic parts. A design tool which makes it easy to create simple interactions within a VE would help to accomplish this; and it would encourage exploration, as playing 'what if?' scenarios with new interactions would be easy to do and give immediate feedback.

3.4 Novice Practices

After initial research into expert practices, we wanted to look at novice behaviour: both in general settings and in the domain of VR authoring. Generic advice on user interfaces [30, 39] is useful for information on supporting novices, as is advice from the design of authoring tools [21, 33, 36]. This helps to create a generically usable tool, e.g., the user interface must be uncluttered, provide immediate feedback, multiple views of the activity and be consistent. Research specifically conducted on end-user programming has elicited learning barriers to these systems, such as design barriers [27]. Design barriers arise from the fact that design is intrinsically difficult; therefore a system must scaffold user creativity in overcoming the barriers [24, 40]. Other types of barriers were more integrated with the software itself, such as users being unable to find and use functionality, or understand its effects. Experts also have problems with some of these barriers, especially design barriers. Design software, even for experts, should be created to scaffold the design process.

There has also been research conducted specifically on 3D programming by novices [9]. Alice, a 3D graphics programming system for novices, was originally developed as a system for rapidly prototyping VEs. The authors found that users needed lots of help with mathematical concepts, preferred direct manipulation interfaces, and benefited from immediate feedback that the system gave on their designs.

Several studies have also been conducted by members of our research group into how novices design and interact with VEs [4, 48, 23]. We have found that people interact effectively with VR even when they are complete novices at computer use. Observational studies of undergraduate students designing VEs for coursework have been conducted; these include both programmers and non-programmers.

A very useful study [4] was of ten undergraduate students from a humanities background who took part in an Interactive Multimedia course at the University of Cape Town, where they were introduced to various aspects of VR and had to design and partially develop a game. They were divided into three groups, with a researcher observing and helping each group. The course included practical lab sessions and tutorials. Students had to produce a design document and use other design aids. They were taught about aids, such as storyboards, flowcharts, maps, etc. Several interesting points emerged. A major problem for all students was the tendency to think in a linear fashion about interactions, as if they were designing for a film. All interactions depend on user actions, so they are not predefined and the narrative is not linear. However, people had trouble conceptualising interactions that diverted from the planned narrative. They did not consider what might happen when the user of the VE did not conform to their design, and often left out alternative user paths entirely. This was even though they had been taught about interactions and how to specify them in terms of basic programming logic. Using a design document tended to amplify this problem, as designers created a story of the VE by describing its look and feel, characters, etc. They became tied to their stories and unable to visualise alternatives.

When students handed in their design aids, they all produced very linear flowcharts of the narrative, which provided evidence of their linear thinking. However, they did understand the formalism well enough to create accurate charts of the typical narrative flow. We found that students used floorplans or maps naturally and very well.

Based on these observations and our own experiences in authoring VEs, we distilled four general categories of typical interaction programming errors, see Table 1. These are errors that we ourselves made repeatedly, and have observed repeatedly in others.

As we did with research on experts, we had to relate information on novice practices to our constructivist values and apply them in our tool design. Once again, the usefulness of multiplicity was highlighted. Novices generally can benefit from viewing multiple perspectives on a design. Multiple representations can be used in VR design to indicate the extent to which interactions are linear and unconnected. These representations also assist novices in reflecting on designs if they are juxtaposed and synched with each other. If novices can create simple interactions, these can be understood easily

(fostering feelings of control), and the ways in which they connect (both obvious and emergent) can be made apparent by scaffolded representations of the interactions and their context. With simple interactions, first steps in design are easy to take and are not alarming. Immediate feedback on consequences is provided and mistakes can easily be fixed.

Table 1: Typical Interaction Programming Mistakes

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 interactions and the way that they relate	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

The combination of simple atomic interactions, multiple representations and contextual scaffolding should help to address sequencing / logical errors (see Table 1), as the system can inform designers of the programmatic consequences of their designs. In the same way, different visualisations can address the other categories of errors. A floorplan can be used to address spatial errors; 3D prototyping with an end-user perspective (i.e. running the designed VE as an end-user would) combined with timelines for scripted sequences can address timing errors; and combined representations and scaffolding can make novice designers aware of the implicit assumptions that they make.

3.5 Iteration

The final step in the first iteration of our process was to create a prototype based on our constructivist values and the practical information which we had integrated with them. This prototype was shown to members of the target group and feedback gained on its problems and successes.

To recap our design requirements: create a tool that provides an engaging problem-manipulation space. Users can directly manipulate objects and their activities and gain feedback through changes in alternative representations even before the 3D world is populated. This is useful for prototyping and concrete exploration. We wanted to make input to the system as simple as possible, in keeping with the constructive value of simplicity. Therefore, we decided to provide simple trigger-condition-action triads, which we named triggersets. These are described with natural language. For example:

if object x is 5m from object y (trigger), *when object y is in location z AND object z is performing animation 'dance' AND user is pressing key 'p'* (conditions) *then object y starts sound 'help me' AND object x moves towards user* (action).

As can be seen from the above example, quite complex interactions can be programmed using very simple atomic parts.

We decided to provide three different representations of the interaction sequence, besides the 3D window: a floorplan, timelines, and a sequence diagram (modified statechart, similar to flowcharts). The floorplan was chosen because this seems to be the formalism which novices understand and use most intuitively. People often have problems understanding 3D space and the effects of movement in 3D, and a simpler 2D representation helps to clarify the effects of acting in 3D [43]. This was also our experience from our observational studies. A floorplan helps to disentangle spatial errors and can be used to graphically place or define objects, waypoints for movement, facing of objects, locations within the VE, etc. To scaffold usage of space in the 3D world, we provide 2D representations for spatial tasks which are not visualised in the 3D world. For example, waypoints provide 2D points (i.e. no height indicated) towards which object can move and locations provide 2D areas of space which can be used for positional triggers (as shown in the example above). Figure 1 displays the 3D window juxtaposed with the floorplan.

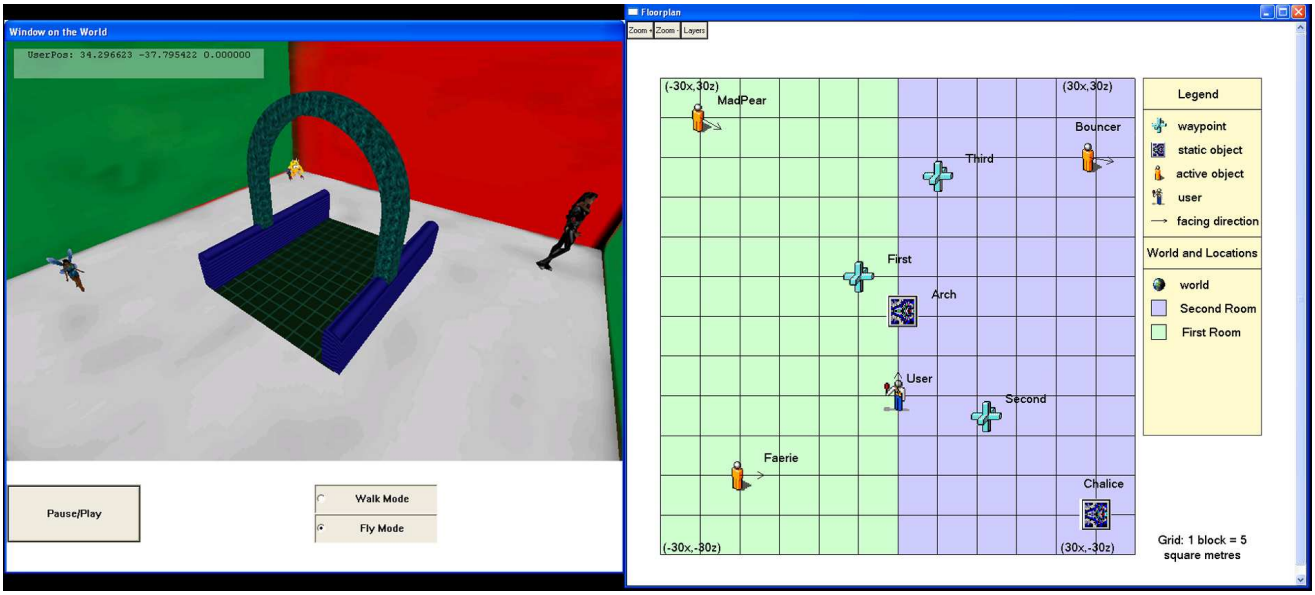

Figure 1: 3D window juxtaposed with floorplan showing different perspectives on design

Timelines are also a well understood formalism. Most of the tools that we examined (Section 3.2) had some form of timeline for sequencing actions. VE design is somewhat different to the design of other media because the actions depend on the user, however VEs usually include sequences where the actions are predictable. For example, during a conversation, the designer may want to synch actor animations and the sound of the conversation, so that when the actor says 'hello', he nods his head at the user. These synched interactions are very frustrating to capture and require repeated running of a VE in order to perfect the timing, as we found in our own VE design experiences. By providing a timeline for these sequences, we make this process much easier. Multiple actions of an actor can be synchronised. In addition, interactions are captured as multiple objects can be placed on one timeline. In order to address the value of simplicity, branching timelines are not used. Each timeline portrays a single sequence of action.

Finally, sequence diagrams were chosen to represent the narrative flow of the VE. We were inspired by the statecharts of Harel [22], which were developed for designing complex reactive systems. A VE can be thought of as a system that reacts to the user, and the goal of effective VE action design is to create a complex network of possible interactions between the user and the system. Statecharts are similar to flowcharts but provide various techniques for reducing the complexity of the diagrams, such as allowing states to be encapsulated within other states. Students used flowcharts naturally, although badly, so we were aware that much scaffolding would need to be provided to help users read these diagrams. The sequence diagrams are automatically generated by the system, based on the triggersets which have been set up by the novice designer. The states represent points in the VE from which user interactions will have effects. The arrows are directly linked to triggers and indicate the effects of these triggers executing. This helps to connect the atomic interactions which have been set up by the designer to the narrative sequences which result. With some help, designers can use the sequence diagram to discover logical errors in their designs.

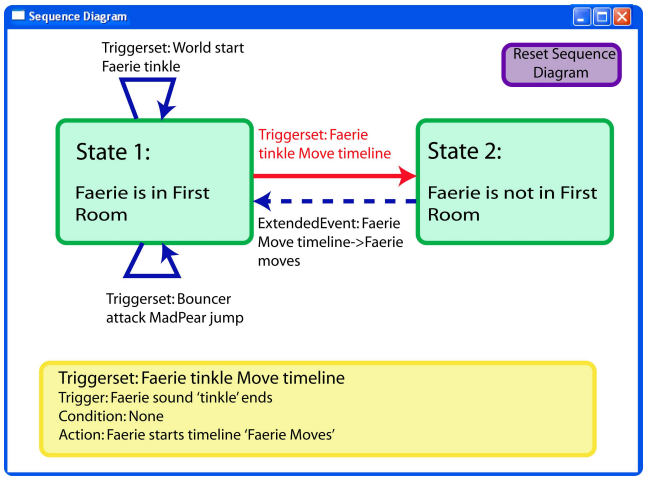

Figure 2: Prototype sequence diagram screen

These three representations interact with the 3D view, which is provided for visualising the result. A run mode of the VE encourages reflection by juxtaposing and synchronising the various representations. It therefore relates the experience of the end-user of the VE to the design.

We created a prototype tool, where the representations were simpler versions of what they would become and the 3D view was not yet available. We did this initial study with our target users to test how the representations and triggerset formalism worked. We had some concerns with how well people could read the formal diagrams and how well they would work with multiple views simultaneously [2]. Figure 2 displays a sequence diagram similar to those used in the study.

Eleven participants were required to examine two existing designs and were given a brief introduction to the tool and half an hour for each examination. The participants all had post-graduate degrees in various disciplines, such as architecture and graphic design. None of them had any graphics programming experience. They had to work out possible sequences of interactions in the first design and find design errors in the second. They were observed during this time and interviewed afterwards. Five of the participants did not receive the representations, as we wanted to examine how effective our diagrams were at helping designers. The results from our study were very encouraging [47]. The subjects who received visualizations worked out the interactions sequences significantly more correctly, and identified twice as many errors as the non-visualization group (although neither group identified very many errors).

Only one participant mentioned difficulty with using multiple views. All of the participants flipped between most of the representations, using them for different aspects of the study.

"*Working with screens to switch between them is easier than looking at triggerset stuff. It is very useful to have visualisations. Also being able to click on them … and work out problem. I had all three open at the same time. Then if you don't understand the sequence you can look at floorplan.*"

"*I used the physical maps first, then triggersets. First I look at all objects, then work from the sequence. Then work out what triggers should go off. I had the sequence, floorplan and triggers all open at the same time - floorplan to see objects, then sequence to move through sequencing, then look at triggersets to back up.*"

All of the participants found the floorplan most useful, as it helped them to orient the physical space. Without a 3D view, this would be necessary.

"*Used floorplan to make environment concrete - very important, really needed it a lot.*"

"*(The floorplan was)…most useful as could see where everything was spatially. Get overview picture in mind. To understand the scene, the map tells you who is in it, where it is.*"

Participants did not use the timelines very much in this study, but as one participant stated:

"*I didn't need to use the timeline as simple things were happening. For design though, I like the timeline. It's important as both a visualisation and a construction tool.*"

Most participants used the sequence diagram well, and found it most useful for finding errors.

"*I used the Sequence Diagram. Most important after the floorplan - only for a reference like a flowchart. It is the least necessary if this is all in the head. The Sequence was useful to check up after your own analysis of the triggers.*"

"*Mistakes were most obvious from the sequence diagram - that many sequences never open - can't open the door, not sure if the jailer return happens. The sequence diagram is good for*

seeing how the triggersets relate, their order and what activates what. I liked highlighting - help to work with the diagram."

We also found some problems during the study. We had viewed the representations as scaffolding for the design process. However, participants wanted more help with using them, especially when it came to finding errors.

"*(Finding errors is)…more confusing to see what is wrong than finding sequence. First was much simpler - maybe less variables. Easier to learn. With the second one, … it is more complicated to understand.*"

Participants also misunderstood the representations occasionally, especially the sequence.

"*Sequence diagram is not completed, as there is nothing for button pressing - basically it stops. I didn't look at the floorplan, that would have told me - I looked for a button press because that is what I saw on the triggersets.*"

This participant decided that the sequence diagram was wrong, as it just stopped, whereas it was indicating that the button press trigger was never executed (which was an introduced design error). We realized that we needed more scaffolding to help people read the representations. In addition, we would need to link them in more obvious ways, as some participants did not think to examine certain representations for clarification when it would have been useful (as with the floorplan quote above).

We are currently completing a second iteration, leading up to a second study, in which we have incorporated all that we had learnt in the first iteration.

4. IMPLICATIONS

In this paper we have described a method for design based on constructivism. This consists of four simple steps, which distil and then apply practical values from constructivism. Expert practices and novice requirements are examined and structured within this framework. This method has been followed within the domain of VR authoring. The quote by Saltzman [35] at the beginning of Section 3.2 states that the difficulty of implementing the mechanics of game designs is easily overlooked in the design process. This also applies to VR authoring. An idea for a design is only as good as its implementation as this is all that the end-user experiences. This design process and the resulting tool should make communication between design partners easier. It should also allow novice designers to work with their own ideas, manipulate them according to the technology constraints and requirements and provide innovative implemented VEs.

We believe that our constructive design method also has general applicability. Design can be applied to anything and each domain brings its own difficulties. There must be freedom built into any generic methodology in order for it to be effective across domains. From the perspective of creativity, it is also preferable to provide the designer with as much flexibility as possible, so that interesting interpretations can emerge. We consider our work to be an application to design tools of Sengers and Gaver's ideas on fostering multiple interpretations in design [38]. Like them, we have tried to specify the usability of our system without controlling its use; we have tried to provide a space for interpretation around VR design; we have fostered multiple opportunities for interpretation by providing multiple representations of the problem. The way in which we differ from the authors is also interesting. While Sengers and Gaver actively resist constraining interpretation in any way, our tool is specifically created to help novices with VE design by providing guiding constraints which could be dispensed with. Without external constraints or guidelines, they would be handicapped by their own lack of knowledge. Therefore, we have tried to encapsulate the expert knowledge that they lack in a way that is easy to learn, while keeping the creative design space as open as possible. Since design itself is so tricky as a concept [12, 37, 41], this flexibility is needed.

This method could also be situated within the paradigm of user-centred design [1], as it focuses on the needs of both the user of the design tool (the designer) and on helping the designer to create an exciting product for the end-user (the player of the designer's VE or game). It departs from this paradigm, in that the users of the design tool may not be experts in the field of design. Therefore, they have to be scaffolded in their use of the tool. Only focussing on the needs of the user has limitations as novice designers are typically not aware of what they might need to design effectively. Constructivism provides more emphasis on supporting novices.

Another paradigm of interest is the Gibsonian ecological approach [19, 20]. It may be regarded as incompatible with constructivism, but we find many points of connection. For example, Gibson defined affordances as the invariants of the environment and objects that the observer perceives and which signify allowable actions. When discussing pictures, he stated that *"Any picture, then, represents what its creator has noticed and considers worth noticing. Even when she paints a fiction or fantasy, she does it with invariants that have been noticed in the course of learning to perceive."* ([19], page 274)

Our premise is that, while expert designers and VE authors are able to perceive the affordances in the environment and objects of a 3D world that allows them to create new designs effectively, novice designers do not know how to perceive them. Therefore, as mentioned above, we chose constructivism because of its focus on supporting learning. We use our multiple representations (with scaffolding in how to read them) to highlight these affordances and simplify the visual display to help novices learn how to perceive the necessary information. Eventually we hope that novices learn through this process to perceive affordances as directly and automatically as the experts whose knowledge is incorporated in the tool. An interesting direction to follow for future research would be to further investigate the ways in which the constructivist and ecological approaches come together.

Our constructive design method (Section 3) specifically lays out the steps to access domain knowledge and the structure within which it can be used in design tools. The process makes space for domain-specific investigations while remaining relevant across domains. All of the constructivist values are generally applicable theoretically. Step 2 of our process makes space in the design process for domain specific investigation of expert practices to apply to the values. Step 3 allows the designer to specifically consider novices in the domain area as well. Therefore, our theory and process provide general guidelines on expert practices and novice problems. They also provide an avenue for domain specific investigation.

5. ACKNOWLEDGMENTS

This research is funded by the South African NRF (GUN 2053402) and by the University of Cape Town. We also wish to thank the anonymous reviewers for their helpful comments.

6. REFERENCES

[1] Abras, C., Maloney-Krichmar, D., and Preece, J. User-Centered Design. In Bainbridge, W. *Encyclopedia of Human-Computer Interaction*. Sage Publications, 2004.

[2] Baldonado, M., Woodruff, A., and Kuchinsky, A. Guidelines for using multiple views in information visualization. In *Proceedings of AVI 2000*, ACM Press.

[3] Bartle, R.A. *Designing Virtual Worlds*. New Riders Pub., 2004.

[4] Beirowski, C., and Vermuelen, H. Experiences with virtual reality accessibility in an African context. In *Workshop proceedings of the IEEE VR 2004 Conference*. 14-18.

[5] Bidwell, N. J., and Holdsworth, J. Battleship by foot: learning by designing a mixed reality game. In *Proceedings of the 3rd Australasian Conference on Interactive Entertainment* (Perth, Australia, December 4-6, 2006). ACM Press, 67-74.

[6] Blackwell, A.F. Diagrams about thoughts about thoughts about diagrams. In *Reasoning with Diagrammatic Representations II: AAAI 1997 Fall Symposium*. 1997, 77–84.

[7] Bowman, D.A. and Hodges, L. Formalizing the design, evaluation and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages and Computing, 10* (1999), 37–53.

[8] Brown, S., Ladeira, I., Winterbottom, C., and Blake, E. The Effects of Mediation in a Storytelling Virtual Environment. In *Springer LNCS 2897 - Virtual Storytelling: Using Virtual Reality Technologies for Storytelling*. 2003.

[9] Conway, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., Deline, R., Durbin, J., Gossweiler, R., Koga, S., Long, C., Mallory, B., Miale, S., Monkaitis, K., Patten, J., Pierce, J., Shochet, J., Staack, D., Stearns, B., Stoakley, R., Sturgill, C., Viega, J.,White, J., Williams, G., and Pausch, R. Alice: lessons learned from building a 3D system for novices. In *CHI Proceedings* (2000).

[10] Cross, N. Creative Thinking by Expert Designers. *The Journal of Design Research, 4, 2* (Nov. 2003).

[11] Do, E. and Gross, M.D. Drawing as a means to design reasoning. In *Visual Representation, Reasoning and Interactions in Design, Artificial Intelligence in Design '96*. Stanford University, 1996.

[12] Dorst, K. On the Problem of Design Problems - problem solving and design expertise. *The Journal of Design Research, 4, 2* (Nov. 2003).

[13] Dow, S., Saponas, T.S., Li, Y., and Landay, J.A. External Representations in Ubiquitous Computing Design and the Implications for Design Tools. In *Designing Interactive Systems* (DIS 06). ACM Press 2006.

[14] Eastman, C. New directions in design cognition: studies of representation and recall. In *Design Knowing and Learning: Cognition in Design Education*. 2000.

[15] Fencott, C. Virtual storytelling as narrative potential: towards an ecology of narrative, In ICVS 2001. 2002, 90–99.

[16] Fosnot, C. T. Constructivism: a psychological theory of learning, In *Constructivism: Theory, Perspectives and Practice* (chapter 2). Teachers College Press, 1996.

[17] Fullerton, T., Chen, J., Santiago, K., Nelson, E., Diamante, V., and Meyers, A. That Cloud Game: Dreaming (and Doing) Innovative Game Design. In *Proceedings of Sandbox Symposium 2006*. ACM Press, 2006.

[18] *Gamasutra: The Art and Business of Making Games*. URL: http://www.gamasutra.com.

[19] Gibson, J.J. *The Ecological Approach to Visual Perception*. Houghton Mifflin Company, 1979.

[20] Gordon, I.E. *Theories of Visual Perception* (3rd Ed). Psychology Press, 2004.

[21] Harada, K., Tanaka, E., Ogawa, R., and Hara, Y. Anecdote: a multimedia storyboarding system with seamless authoring support. In *ACM Multimedia*. 1996, 341–351.

[22] Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., and Trakhtenbrot, M. STATEMATE: a working environment for the development of complex reactive systems. In *IEEE Transactions on Software Development, 16, 4* (1990), 403–414.

[23] Hendricks, Z., Marsden, G., and Blake, E.H. A Meta-Authoring Tool for Specifying Interactions in Virtual Reality Environments. In *Proc. African Graphics Assoc Conf.* (Cape Town, South Africa, 2003). ACM Press, 2003.

[24] Hewett, T.T. Informing the Design of Computer-Based Environments to Support Creativity. *International Journal of Human-Computer Studies, 63, 4-5* (October 2005).

[25] Kaur, K., Sutcliffe, A., and Maiden, N. A design advice tool presenting usability guidance for virtual environments. In *Workshop on User Centred Design and Implementation of Virtual Environments*. York, England, 1999.

[26] Kavakli, M., Suwa, M., Gero, J., and Purcell, T. Sketching interpretation in novice and expert designers. In *Visual and Spatial Reasoning in Design*. Univ Sydney, , 1999, 209-220.

[27] Ko, A.J., Myers, B. A., and Aung, H.H. Six learning barriers in end-user programming systems. In *IEEE Symp Visual Lang & Human Centric Computing*. Rome, 2004, 199-206.

[28] Laurel, B. *Computers as theatre*. Addison-Wesley Publishing Co., Massachusetts, 1993.

[29] Leão, L. The labyrinth as a model of complexity: the semiotics of hypermedia. In *Computational Semiotics for New Media (COSIGN '02)*. September 2002.

[30] Norman, D. *The psychology of everyday things*. Harper Collins Publishers, USA, 1988.

[31] Pearce, C. and Ashmore, C. Principles of Emergent Design in Online Games. In *Proceedings of Sandbox Symposium '07*. ACM Press, 2007, 65-71.

[32] Petre, M. and Blackwell, A.F. Mental imagery in program design and visual programming. In *International Journal of Human-Computer Studies, 51* (1999), 7-30.

[33] Repenning, A., Ioannidou, A., and Phillips, J. Collaborative use and design of interactive simulations. In *Proc ACM Conf CSCL*. ACM, 1999.

[34] Salovaara, A., Johnson, M., Toiskallio, K., Tiitta, S., and Turpeinen, M. Playmakers in Multiplayer Game Communities: Their Importance and Motivations for

Participation. *ACM SIGCHI Int Conf on Advances in Computer Entertainment Technologies (ACE2005)*. ACM, 2005.

[35] Saltzman, M. *Game Creation and Careers: Insider Secrets from Industry Experts*. New Riders, 2003.

[36] Schiphorst, T., Calvert, T., Lee, C., Welman, C., and Gaudet, S. Tools for interaction with the creative process of composition. In *CHI '90* Proceedings. ACM Press, 1990.

[37] Schön, D.A. *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in the Professions*. Jossey-Bass Inc, 1987.

[38] Sengers, P. and Gaver, B. Staying Open to Interpretation: Engaging Multiple Meanings in Design and Evaluation. *Designing Interactive Systems (DIS '06)*. ACM, 2006.

[39] Shneiderman, B. *Designing the user interface: strategies for effective human-computer interaction (2nd edition)*. Addison-Wesley Publishing Co., Massachusetts, 1992.

[40] Shneiderman, B. Creating Creativity: User interfaces for Supporting Innovation. In *ACM Transactions on Computer-Human Interaction, 7, 1* (March 2000), 114 – 118.

[41] Simon, H. A. The Structure of Ill Structured Problems. *Artificial Intelligence* 4 (1973), 181-201.

[42] Spiro, R.J., Feltovich, P.J., Jacobson, M.J., and Coulson, R.L. Cognitive flexibility, constructivism and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. In *Constructivism and the Technology of Instruction*. Erlbaum, 1992.

[43] Tufte, E. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, UK, 1983.

[44] *Virtools - The Behaviour Company*. URL: http://www.virtools.com.

[45] von Glasersfeld, E. Introduction: aspects of constructivism. In *Constructivism: Theory, Perspectives and Practice*. Teachers College Press, 1996.

[46] Wilson, B. and Lowry, M. Constructivist Learning on the Web. In *Learning Technologies: Reflective and Strategic Thinking*. Jossey-Bass, San Francisco, 2001.

[47] Winterbottom, C., Gain, J., and Blake, E. Using Visualizations to Support Design and Debugging in Virtual Reality**.** In *Proceedings of the International Symposium on Visual Computing (ISVC '06)*. Springer-Verlag, 2006, 465-474.

[48] Yang, S and Marsden, G. Eliminating Design and Execute Modes from Virtual Environment Editors. In *Proceedings Presence 2004* (Valencia, Spain), 2004.