

Error Handling in Speech User Interfaces in the Context of Virtual Worlds

Markku Turunen

University of Tampere
Department of Computer Science
Kehruukoulunkatu 1
33101 Tampere
mturunen@cs.uta.fi

ABSTRACT

While the virtual environments are gaining more success each day there are many usability problems associated with them. The controlling methods, which usually rely heavily on the user's hands, are one problem. There are situations and user groups in which this is not an acceptable way to use the computer. Here we approach this problem by using speech input.

The relatively high error rates are the main problem when speech input is used. This problem is even worse when speech is used in situations where speed is an important factor. This all means that advanced error-handling methods must be applied.

We studied different errors that could occur in 3D worlds by constructing a speech-enabled VRML 2.0 browser. In addition to speech recognition errors we investigated errors that were caused by the interaction metaphor used. Based on the informal user tests we introduce some improved error handling methods.

Keywords

3D user interfaces, speech user interfaces, speech recognition, error handling, VRML, virtual worlds

INTRODUCTION

Three-dimensional user interfaces are becoming more common all the time. One of the main reasons for the popularity of these interfaces is the expanded use of virtual reality. It is believed that virtual reality produces the same kind of effect that graphical direct manipulation interfaces did in the 80's.

However, human computer interaction is more complex in three-dimensional space than in the regular two-dimensional desktop environment. One reason for this complexity is the nature of the three-dimensional environment. Another factor is the inadequacy of the interaction methods. Research has approached this challenge mainly by producing new high-tech input devices and powerful three-dimensional widgets [2]. A common element for both of these approaches is that they require keen motor control and rely heavily on the user's hands.

Although these approaches allow efficient interaction methods for most cases, there are always situations when the user's hands are occupied and therefore not available. This kind of situation could be, for example, a task where the user controls other devices or programs at the same time. There are also people who have disabilities that prevent them from using controllers that require precise kinetic movements. Alternative methods are needed so that these all the time more common three-dimensional interfaces could be available in all situations and to all users.

Speech is one solution to this problem. Speech is a natural and universal way to communicate. The use of speech does not require any kind of special equipment and it is always available. Speech could be used even in situations where the computer is not physically accessible. It has been predicted that speech would be the dominating factor in the user interfaces of the future [3].

To examine the usefulness of speech as an input method in three-dimensional virtual environments we built a speech-enabled virtual world browser. This browser allows full control of those virtual worlds that follow the VRML 2.0 standard. VRML was chosen because of its popularity at the World Wide Web.

Problems in using speech as an input method

The use of speech as an interface element is not very common yet. The high error rates are one reason for this. This is mainly because of the immature speech recognition technology. It has been said that the technology controls the designing of speech user interfaces [7]. Therefore error handling is the main factor in the designing of speech user interfaces [6]. We investigate here recognition errors and solutions that are important in real-time user interfaces.

Because of the complexity of the interaction and the ambiguous nature of the presentation of the three-dimensional space on a two-dimensional screen it is not always clear how to control a virtual world. The users often make incorrect actions which are not meaningful in the current situation. Here we study these *semantic errors* from the viewpoint of the *interaction metaphors*.

To investigate the different kind of errors and their effects we carried out an experiment, a set of tests where the users

navigated in the virtual worlds by using our browser. As a result of this experiment we will present here several advanced error handling methods.

This paper is organized as follows. First we will discuss the speech recognition errors. Then we consider the semantic errors that are caused by the interaction metaphor and the ambiguous nature of a 3D environment. Previous research about error handling will be discussed next. Our speech enabled VRML-browser is briefly detailed after that. Then follows a description of the main findings from the user tests. Based on the results of the user tests and the previous research we will introduce some advanced methods for error handling.

SPEECH RECOGNITION ERRORS

From the user's point of view, there are three types of recognition errors: *deletions*, *insertions* and *substitutions*. The deletions are usually caused by words that the recognizer does not understand at all. The insertions are mainly caused by the circumstances and the equipment used. The substitutions occur when the recognizer cannot reliably understand what the user means.

The performance of the speech recognition system could be expressed as *word error rate* [13]:

$$\text{word error rate} = \frac{\text{substitutions} + \text{insertions} + \text{deletions}}{\text{number of words spoken}} \star 100$$

The word error rate (error rate in short) indicates how many errors the recognizer produces compared to all spoken words. It is a standard method for comparing and evaluating speech recognizers, but to be meaningful it should only be used to compare tasks which are at the same level. The task complexity is measured by the *perplexity*, which could be understood as the result of the possible word combinations in the system's vocabulary.

The current state-of-the-art systems could achieve error rates as low as 0.3% in tasks which have very low perplexity, like in digit recognition. On the other hand high perplexity tasks, like telephone based conversational systems, could hardly produce error rates below 50% [13]. Because our speech-enabled browser belongs to the group of the low perplexity tasks we could expect relatively low error rates. But even in the low perplexity tasks it is important to try to cut down error rates so that user satisfaction is achieved.

SEMANTIC ERRORS

Semantic errors are commands that are either illegal or not meaningful in the current context. Here we are interested only in the latter, because they relate more to the 3D features of the interaction methods. There are two main sources for these semantic errors: the user either misinterprets the 3D scene or does not understand the consequences of his/her commands.

If the user misinterprets the presentation of the virtual world it is very likely that the misinterpretation is caused either by insufficient *depth cues* or failures in the user's *spatial rea-*

soning. The depth cues, which include motion, perspective, shadows etc. are elements that help us to realize the 3D nature of the scene. Spatial reasoning is a mental process that involves thinking about the relations between the three-dimensional objects [9].

Here we are not interested in depth cues as a subject; they are actively studied elsewhere and are not specific to speech input. Instead, we use the depth cues in our study to investigate other aspects. First, we constructed our test cases so that the user had always the necessary depth cues available. Second, by using the depth cues we can improve the user's spatial reasoning ability.

We used transparency to support the user's spatial reasoning so that we could separate the semantic errors which are caused by spatial reasoning from those which are caused by the *interaction metaphor* used. We chose transparency because it has been successfully used before for target acquisition [12], which motivated us to use it in our study.

When the user understands the relations between the objects and therefore knows how to act in a three-dimensional scene but still makes mistakes, then the reason could be that the user does not understand the control metaphor used, so he/she does not understand what are the consequences of his/her actions. The control metaphor is a model that expresses how the user's commands are mapped onto movements. If the metaphor does not match the user's internal model then the user's commands are likely to be misunderstood by the system.

Here we focus on the semantic errors that are caused by the interaction metaphor because they relate more than the other aspects to the use of speech as input method. Our goal is to try to understand how and in what situations these semantic errors are likely to occur and how they affect the interaction.

ERROR HANDLING

Most of the existing speech interfaces are not real time in the strict sense: delays in the computer's responses could be as much as seconds. This could not be acceptable at the virtual environments because even brief delays could be very destructive for user satisfaction [8]. It is obvious that all interactions, including error handling, should be very fast and painless to the user.

We can find five different phases in the error handling procedure. They are *error prevention*, *error detection*, *diagnosing the cause of error*, *designing error repair* and *executing error repair*.

Error prevention deals traditionally with things like using a switch so that the users can turn off the recognition when they are not communicating with the computer etc. Another commonly used error prevention method is rejection based filtering where a command is ignored if its recognition value is below a certain *threshold value*. In this way we could reduce the substitutions and the insertions. We are also interested in methods that could prevent semantic er-

rors, especially those caused by the interaction metaphor. To approach that we introduce later in this paper several methods to support the interaction metaphor.

In order to correct the errors we must first detect them. Usually error detection is left for the user. However, this is not a very optimal approach. All errors are not perceived in this way. The user must also keep his/her attention in this process. Therefore automatic error detection methods are required. Here we introduce *error correction sequences* as one solution to find out when the errors are likely to happen.

Error correction should not be started without careful planning. First we should analyze the reasons of the errors. Usually this is very difficult because of variations in speech signal and high need for the context information in the case of semantic errors. Because the reasons of the recognition errors are unreachable, we focus here on the semantic errors. We are especially interested in knowing what proportion of errors are caused by the interaction metaphor.

When the reasons are analyzed both the further consequences of the error and the repair costs of computer supported error correction must be carefully compared. If the costs of error correction exceed the costs that the error causes then there is no point to start the error correction procedure. One can use error correction sequences to measure what are the costs of the user performed error correction and thus the consequences of the error.

Finally, if error repair is a worthwhile choice, it could be performed in a variety of ways. The most common method is to repeat the original input. This is the easiest, and in many situations like in the case of deletion errors, the only way to correct the errors. There are also other methods such as selecting from a list of choices, paraphrasing the input or using some other modality if the user interface is multimodal [7].

Because the costs of manual error correction are usually high, we must try to find methods that are easier for the user. One alternative is confirmations. They are not actual correction procedures because they do not repair the consequences of the error; instead they prevent the consequences from occurring. Confirmations could be described as something between error prevention and correction.

In [10] confirmations are divided into implicit and explicit confirmations. Explicit confirmations are more like manual error correction and therefore bulky because the user always has to respond. They are a preferred method in real-time situations only when the consequences of the errors are hard or impossible to repair.

Implicit confirmations are a much better choice when speed is an essential factor. When used in conversational systems, the implicit confirmation procedure informs the user that an error has occurred and waits if the user wants to cancel the action or not. If the user does not cancel the action it is carried out, otherwise the input could be ignored or the error

correction procedure could be started. Here we are interested in how we could reduce the consequences of the errors by using the implicit confirmations.

To investigate different errors and their effects we built an experimental VRML 2.0 browser, which uses speech to control virtual worlds. The browser's functionality and implementation are described in detail in [11]. Here we present only the browser's basic concepts that are needed to understand the user tests.

SPEECH-ENABLED VRML BROWSER

The speech-enabled VRML browser allows the virtual world's basic movements to be carried out by using speech. To maximize the recognizer's accuracy only a small set of commands was used. The browser was developed on a Sun UltraSparc 2 under Solaris 2.5.1. Our implementation is based on an existing VRML browser [1] and on a commercial speech recognition system [4]. We present here the browser's functionality and the error prevention method used.

Functionality

We gave the user a possibility to move to all four directions, zoom closer to the scene or draw away from the scene. These actions were implemented as direct viewpoint placement movements according to the basic axis. This straightforward model was chosen because it allowed unlimited movement. It is also the basic interaction mechanism in most VRML browsers. Figure 1 illustrates these *moving commands*.

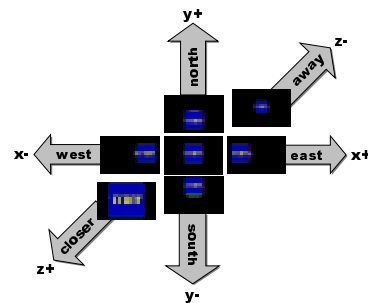


Figure 1: Moving commands.

To allow the user to examine the objects we implemented a model in which the user moves in a circle around the object placed in the center of the world. This model is not very usable at realistic worlds, like city models, but for our purposes it offered many benefits. First, this model is very powerful when examining the objects from different viewpoints. The user feels like he/she is at a virtual museum where he/she could freely move in the space and around the objects. Second, this model allowed us to perform various kind of positioning tasks, in both 2D and 3D situations. This model is also familiar from the existing browsers. Figure 2 illustrates these *turning commands*.

All actions are carried out from the user's point of view. This means that we used the "eyeball in hand" –metaphor as

opposite to the “scene in hand” –metaphor. The “eyeball in hand” –metaphor was chosen because it suits better for movement and the first timers adopt it easily. For the detailed explanations of control metaphors and their usage see [15].

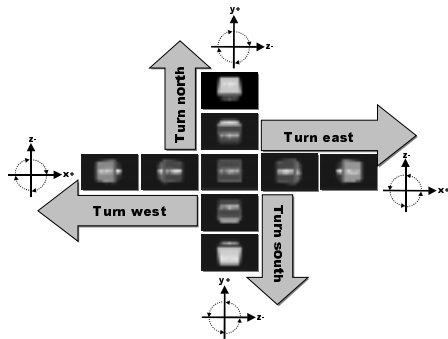


Figure 2: Turning commands.

We offered two modes for controlling the movement. In the *continuous mode* the user’s commands are executed until he/she gives another command or uses the “Cut” command to stop the movement. In the *constant mode* the user moves only a certain distance.

Error filtering

We implemented a basic filtering method to reduce substitution and insertion errors. In this filtering some commands are rejected because of their low *recognition value*. The recognition value is a numeric value, which expresses how certain the interpretation of the recognizer is. When the recognition value is below the so-called *threshold value*, it is rejected. This means in practice that some of the substitution and insertion errors are transformed into deletion errors during the interaction.

We presumed that filtering would increase the user’s satisfaction because both the substitutions and the insertions are more devastating than the deletions. This is true in most speech user interfaces, but in the real time user interfaces this could not always be the case. The user studies showed that this kind of too simplistic filtering is in reality more hindering than supporting.

INFORMAL USER TESTS

We carried out sixteen informal user test sessions to investigate what kinds of errors occur during the speech control of virtual worlds and what are the consequences of these errors. We investigated both recognition errors and semantic errors. Because we were interested in the semantic errors that are caused by the interaction metaphor, we followed how the users adapted to the interaction metaphor. Based on the findings from the tests we introduce some advanced methods to handle the errors.

All test users were experts in computer use and most of them had only a few experiences with virtual worlds. The tasks used in the test sessions are introduced briefly below.

Basic movement task

In the basic movement task (see Figure 3) the user was asked to visit the squares so that the cursor was upon the target. We used here the continuous mode and the user needed only the moving commands.



Figure 3: Basic movement task.

Each user was presented twenty targets and the targets were the same for every user. The targets were presented one at a time. We used paper and pen to point out the targets.

In the basic movement task we wanted to find out how the users adopted to the control metaphor and, like in all tasks, what kinds of errors occurred and how they affected the interaction.

Turning tasks

Two worlds were used in the turning tasks. In the first world (Figure 4, at left) the user was presented with a color cube, which was constructed in a way that allows the user to see all of the cube’s six sides at once regardless of his position. This was supposed to simulate the ultimate transparency effect. The second world (Figure 4, at right) used the cube of same size and shape, but this time the cube was filled so the user could only see one side at a time when the cube was presented in its original position perpendicular to the screen.

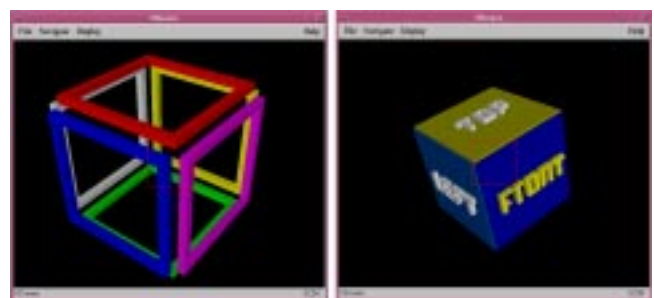


Figure 4: Turning tasks.

The user’s task in these worlds was to move to a particular side that was pointed to him/her. Like in the basic movement task, there were twenty targets. The constant mode was used to control the movement and the presentation method was the same as in the basic movement task.

In the turning tasks we investigated how the control metaphor suited object examination. Second, we were interested

in finding out how many semantic errors were caused by the user's misinterpretation of his/her position in the world.

Our main goal here was to try to understand how greatly the user's spatial reasoning abilities affect the navigation. By using transparency we were able to separate these from the semantic errors that result from difficulties in adopting the control metaphor.

Positioning task

In the positioning task (Figure 5) the user was instructed to position himself/herself so that the cube and the ball were perpendicular to him/her and the cube filled the cursor. There were five different worlds, in which the cube and the ball were placed in different positions and orientations.

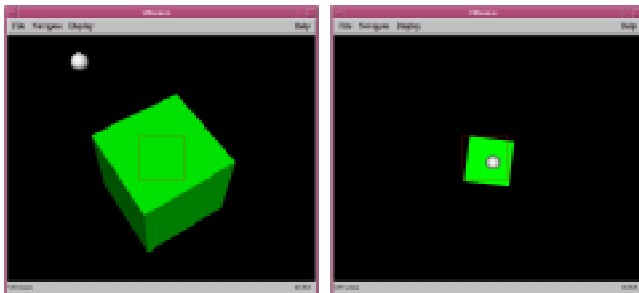


Figure 5: Example of the positioning task.

The positioning task was basically the same as the basic movement task combined with the first turning task. Because of the presence and the placement of the ball the user could easily figure out the cube's orientation.

Here we wanted to find out how the user's ability to control varied because of the increased complexity of the world. It is likely that the user felt that he/she was controlling the cube and the ball, not moving around them. The situation was the same in the turning tasks. Because of the real-world nature of the objects in the turning tasks there existed reasons to believe that it was the turning tasks rather than the positioning tasks where the users felt that they were manipulating the cube. Our goal was to find out if this was true and how people handled these situations.

RESULTS AND CONCLUSIONS FROM TESTS

As results of our informal tests we present here some key findings that concern recognition errors and semantic errors. Based on these findings we introduce some improved error handling methods in the following section. The results presented here are very preliminary. They should be further investigated and verified using formal experiments.

Recognition errors

Table 1 presents recognition errors and how they are divided among the different error types. The total error rate was around 14%. Because the perplexity of our browser was low, the error rate was quite high. This is mainly because of the environment, which was an office room in a not-so-quiet environment. The error rate was very suitable

for our purposes because it allowed us to investigate different errors in real situations without being too high.

	insertions	deletions	substitutions
relation to commands given	0.3 %	10 %	4 %
relation to errors	2 %	74 %	24 %

Table 1: Recognition errors.

As Table 1 shows, deletions are the dominating error type. We analyzed the deletion errors further and found out that the filtering process caused 30% of them. The rest were just ignored by the speech recognizer. Now our interest was to find out how many substitutions were correctly rejected. The result was a big disappointment: for every correct rejection the system ignored five correctly recognized commands. It is easily concluded that our filtering procedure was very unsuccessful.

We collected all recognition values that the recognizer produced during the test sessions. As we looked at those we noticed that the incorrect use of filtering was not the main reason for the incorrect deletions: the recognition values were so widespread that it was impossible to make any reliable assumptions from them. So we conclude that it is impossible to filter out substitutions only from the speech recognizer's information and there exists strong need to include context information into filtering.

Another interesting observation concerns user satisfaction. Most users told that the deletion errors are the most annoying ones. Obviously one reason for this is their high proportion, but it is unlikely that this is the only reason. We argue that it is useful to try to balance the different error types in order to achieve maximal user satisfaction. This approach is also beneficial from the viewpoint of the error correction costs, as we would see in the following section.

Semantic errors

Figure 6 shows the semantic error rates in the different tasks. As can be seen, the semantic error rates vary a lot between the tasks. They were very low in the basic moving task and in the first turning task but then increased greatly.

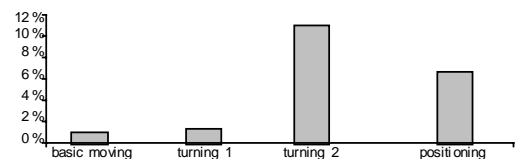


Figure 6: Semantic errors.

We asked the users how they felt about the first two tasks, where the error rates were low. All users responded after the first task that interacting with the world was quite natural. But when we asked it again after the second task, most of them replied that they did not initially like the way the commands were carried out. This suggests that they have different internal metaphors. But as the results indicate, they are still able to perform the task surprisingly well. The

first conclusion that we could draw from the semantic errors was that the users adopted the control metaphor very well, even if it did not match the one they initially had.

As we examined the semantic errors in the second turning task, we found that the error rate had increased dramatically. As the task was exactly the same as the previous one but now without the help of transparency, we could conclude that the errors were because of the misinterpretation of the 3D scene, not because of the interaction metaphor. This indicates that in the basic situations the interaction metaphor is a less important factor than the spatial reasoning.

The results from the fourth task are the most interesting ones. As could be seen, the error rates were almost as high as in the third task. As the task was basically the same as the first two tasks it was obvious that the errors were caused by the interaction metaphor. We asked again how the users felt about the control. The users mainly experienced that they were controlling the objects. This was like in the second and the third task. But unlike in those tasks, the users did not perform very well.

The findings from the fourth task suggest that the users had difficulties in keeping the control metaphor. We can conclude that in situations where two kinds of tasks are mixed the users cannot keep their mental model the same as the control metaphor. This in turn suggests that the interaction metaphor needs more support.

IMPROVED ERROR HANDLING

Based on the observations we made during the user tests, we present here some improved error handling methods. First we introduce the error correction sequences, which are the basis of many other error handling methods. Then we suggest some ways to prevent both recognition errors and semantic errors. Finally we show how the interface designers could handle the actual error corrections.

Consequences of errors

As we have seen before, to effectively detect and correct the errors we need information about the context where the errors occur. Here we take an approach where the context of the errors is presented as sequences of correcting actions. We can use these *error correction sequences* to both detect errors and measure the costs of user performed error corrections.

The actions needed to correct the errors are very context and error type dependent. In order to analyze different configurations we divide the situations based on the current state: if the user is moving or not. Table 2 shows the different error correction sequences when the user is not moving and Table 3 presents the sequences when the user is moving.

The abbreviations used in the error correction sequences are as follows: *C* stands for the correcting command, *S* for the stopping command, *R* for repeating the erroneous command and *L* for repeating last command before the erroneous

command. The parenthesized commands are not needed necessarily, but most people use these commands. It is important to notice that the costs of the commands are dissimilar. As an example, in repeating the erroneous command it is not very uncommon that the same error happens again. Also the users do not like at all to repeat commands.

Error type	Constant mode	Continuous mode
1. Insertion	C	(S) + C + S
2. Substitution	C + R	(S) + C + (S) + R
3. Deletion	R	R
4. Semantic	C + C	(S) + C + (S) + C

Table 2: Error correction sequences when not moving.

The first use of the error correction sequences is the automatic detection of the erroneous commands. This allows the computer to notice that an error has occurred. As we have stated before, automatic error detection is needed because it decreases the burden of the user and helps the recognizer to estimate its performance.

Error type	'Cut'	Other command
1. Insertion	L	(S) + C + (S) + L
2. Substitution	(R) + C + R	R (to 'Cut')
		(S) + C + (S) + R (to other)
3. Deletion	(R) + C + R	(S) + C + (S) + R
4. Semantic	L	(S) + C + (S) + C

Table 3: Error correction sequences when moving.

By using correct parameters in the error correction sequences we could calculate the costs of the error corrections and therefore measure the total efficiency of the system. This requires that we could estimate the amount and the distribution of the different errors. We could also use these measures to determine when it is worthwhile to start automatic or interactive corrections and when it is advantageous to let the user to correct the errors.

Preventing errors

As mentioned earlier, the first phase in error handling is error prevention. By preventing the user from producing errors we could increase the system's usability. Here we introduce two kinds of error prevention methods for both recognition errors and semantic errors.

Filtering incorrect recognitions

As our initial filtering method failed, we planned our new method to be based on the context information obtained from the environment. Most recognizers produce alternative hypotheses. Without any context information the alternative hypotheses are meaningless – one can only guess which of them is the correct one. But when the context information is introduced we could pick the hypothesis that makes the best overall score when merged with the context information.

The problem with this method is: where can we obtain that context information? This question leads us close to the unsolved problem of artificial intelligence, because we have to understand what the user wants to do. Fortunately, we do not have to understand all the user's meanings, because it could be easily reasoned in most situations which commands are irrelevant and which are likely to happen. As an easy example: if the user is moving toward a target and he/she misses it by a couple of inches and gives a command that could be interpreted as a command toward the target or a command away from the target, it is very likely that the first interpretation is correct. Of course this is not true in every situation and most situations are much more complex.

We can also use context knowledge obtained from the error correction sequences to support the recognition. If it is useful to let the user to correct the errors, the computer could support this by giving better recognition values to the commands that are likely to occur. This prevents new errors from happening. This is an important aspect because new errors are common in error corrections.

Supporting the control metaphor to prevent semantic errors

As we concluded from the user tests, in simple tasks the users adopt the control metaphor very well. For those tasks it may be more useful to focus on other aspects like giving the user more information about the scene using depth cues etc. But when the situations are complex additional methods are needed.

We introduce here the *context-sensitive cues* as our solution to support the metaphor. By context-sensitive cues we mean hints that are placed into the scene so that they express what actions the possible commands could produce.

The context-sensitive cues could be implemented in a variety of ways. The simplest way is to express the names of the commands as labels and put them into the scene around the user's current position. For the movement commands this is very simple because the labels lie on the plane. For the other commands different kind of positions could be calculated and the optimal ones could be tried. There are many algorithms available for this kind of optimization.

To improve the textual cues different symbols could be used. We give here one example, which is based on the existing three-dimensional widget known as the interactive shadows [5].

The interactive shadows are a combination of widgets and depth cues at the same time. The basic idea is to present the shadows of the objects and allow the user to manipulate them. Because the shadows lie on the plane they are restricted in a way that makes them to be two-dimensional controllers. The idea's adaptation to our needs is very straightforward: the shadows are constructed so that they express both the objects and the command words.

Another way to support the interaction metaphor could be for example the use of animation and symbols to give the user a stronger feeling about what he/she is doing and how

he/she is controlling the world. Sometimes it could be useful to allow the user to change the metaphor. This could also be automatic, but if it fails it is likely that the user becomes disoriented with the scene.

If the speech recognition system supports dynamic vocabularies, it could be very useful to allow the user to construct a command word set of his/her own. As one of our test users put it: "It would be nice to have my own set of commands. Now I feel like I'm using someone else's words". If the user selects the command words, it is likely that the formed interaction metaphor will be much stronger.

Error correction based on implicit confirmations

As stated before, error correction should be very slight when used in real-time applications. The explicit error correction methods could destroy the real-time nature of the application so the implicit methods are needed. We based our approach to the implicit confirmation, which could be used in conjunction with a canceling command. In this way the user is able to correct the errors smoothly.

When a possible error is detected, the first step in the error correction process is the presentation of the error information. This could be done in several ways. We prefer the context-sensitive cues presented earlier in this paper. They can express both what happens if the user cancels the action and what happens if he/she decides to continue.

Usually the implicit confirmation waits for the user to cancel the action and if nothing is going to happen for some time the interpreted command is executed. This could not be the case in the virtual environments. The action must be executed immediately and the implicit confirmation process must be started at the same time. Our implicit confirmations are like intelligent "undo" commands that do not leave the user alone to start the error correction.

If the user decides to cancel the action, three different approaches are possible. According to the first, the motion is stopped and the user is let to decide what happens next. In some cases it could be advantageous to let the user give the correcting commands. We could use the error correction sequences to measure the costs of the manual corrections when choosing between the automatic and the user performed corrections.

The second approach uses a mechanism where all consequences of the last command are eliminated and the user is returned to the situation before the erroneous command. All this could be done automatically and without any user involvement. This could be the most useful approach when the consequences of the erroneous command are hard to correct manually. The automatic correction, or explicitly given "undo" command, is not always easy to perform. This is especially true in the dynamic virtual worlds. The implementation of the intelligent undo-operation is therefore a very challenging task.

The third approach suits situations where it is hard to decide how to proceed. One example of this is the situation

where the user is near the target and the system could not decide which of the possible commands is the right one. In these situations the best way to proceed is to start the explicit error correction procedure like letting the user choose the right command from the list of possible values.

CONCLUSIONS AND FUTURE WORK

The user tests showed that there exists strong need for both the improved error prevention and the error correction. The results also indicate that these actions are highly context dependent and should not be used without careful planning.

To detect the errors and to measure the costs of the user performed error correction we introduced the error corrections sequences. Based on these we could measure the system's total efficiency and decide when it is useful to start the automatic or the computer supported error corrections. We could also use this information to support the recognition. In follow-up studies we will verify how these sequences work in real situations and what are the parameters of the different correcting actions.

Our initial recognition error filtering was very unsuccessful. To correct the situation, we suggest that improved methods should be based on the context information. In the future we will implement different decision functions to choose the right commands from the list of alternatives. We will be focusing on the general decision functions that are suitable for most virtual worlds.

The results indicate that the control metaphor used does not have a notable effect in the simplistic situations because the users could adapt to the metaphor very well. However, in the case of complex situations there is a strong need to support the metaphor. We introduced here the context-sensitive cues as one solution. In the future we will be implementing some example cues, like the one mentioned earlier and the one based on the moving motion metaphors [14].

Error correction can be very annoying and destructive if it is designed poorly. We presented here the error correction mechanism, which is based on the implicit confirmations. In the future we will implement our model and evaluate its usefulness. We will also implement the intelligent undo command.

REFERENCES

1. Andrews, K., Pesendorfer, A., Pichler, M., Wagenbrunn, K.H., Wolte, J. Looking Inside Vrwave: The Architecture and Interface of the Vrwave VRML97 Browser. In *Proc. of VRML'98*, Monterey, California, Feb. 1998.
2. Conner, B., Snibbe, S., Robbins, D., Zeleznik, R., van Dam, A. Three-dimensional widgets. In *Symposium proceedings on Interactive 3D graphics*, 1992: 183-188. ACM.
3. van Dam, A. Post-WIMP User Interfaces. *Communications of the ACM*, 40 (2): 63-67.
4. graphVite Speech Recognition Prototyping System. Entropic Corporation.
[<http://www.entropic.com/htk/graphvite.html>]
5. Herndon, K., Zeleznik, R., Robbins, D., Conner, D., Snibbe, S., van Dam, A. Interactive Shadows. In *Proc. ACM SIGGRAPH Symposium on User Interface Software and Technology*, 1992: 1-6.
6. Kamm, C. User Interfaces for Voice Applications. In *Voice Communication Between Humans and Machines*. Roe, D., Wilpon, J. (editors). National Academy Press, Washington D.C., 1994: 422-442.
7. Mane, A., Boyce, S., Karis, D., Yankelovich, N. Designing the User Interface for Speech Recognition Applications. *SIGCHI Bulletin*, 1996, Volume 28 (4): 29-34.
8. Mine, M. Virtual Environment Interaction Techniques. UNC Chapel Hill Computer Science Technical Report TR95-018, 1995.
9. Osborn, J., Agogino, A. An Interface for Interactive Spatial Reasoning and Visualization. In *Proc. ACM CHI'92 Human Factors in Computing Systems Conference, ACM SIGCHI 1992*: 75-82.
10. Schmandt, C. *Voice Communication with Computers*. Van Nostrand Reinhold, New York, 1994.
11. Turunen, M. Puheohjaus 3D-käyttöliittymissä. Master's Thesis, Department of Computer Science, University of Tampere, 1998 (in Finnish).
12. Zhai, S., Buxton, W., Milgram, P. The partial-occlusion effect: Utilizing semitransparency in 3D human-computer interaction. *ACM Transactions on Computer-Human Interaction*, 3(3): 254-284. 1996.
13. Zue, V., Cole, R., Ward, W. Speech Recognition. In *Survey of the State of the Art in Human Language Technology*. 1996.
[<http://www.cse.ogi.edu/CSLU/HLTsurvey/HLTsurvey.html>]
14. Ware, C. Moving Motion Metaphors. In *Proceedings of CHI '96: ACM conference on Human Factors in Computing Systems*: 225-226. New York, NY.
15. Ware, C., Osborne, S. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. In *Proceedings of 1990 Symposium on Interactive 3D Graphics*, ACM SIGGRAPH, 1990: 175-183.