

avatar, helps user navigate in Second Life. It can also be used to query the environment through a chat-like interface. Nevertheless, both the IBM's Virtual Worlds User Interface for the Blind and the virtual guide dog lack the functionality to provide detailed descriptions of the virtual environment.

TextSL [10] is a command-based virtual world interface for users with VI. TextSL allows users to communicate with other avatars, navigate in the environment and explore the new world with simple commands. It iteratively accepts typed commands and creates textual feedback which can be used by screen readers. It is a web interface which allows access from any device having a web browser (including mobile devices). It also has a speech synthesizer of its own which eliminates the dependency on a system having a screen reader while users still have the option to use one. TextSL communicates with virtual worlds using an open-source library called the openmetaverse [6]. This library allows the interface to be used to access any openmetaverse-based virtual world with no additional effort.

The interaction of TextSL is inspired by multi user dungeon games (e.g. Zork) as virtual worlds have their origins in such games and the way input and output is provided is suitable to be read with a screen reader or tactile display. TextSL commands can be grouped into four categories: 1) **Exploration**: move, fly, follow, teleport, describe; 2) **Communication**: say, whisper, mute, shout; 3) **Interaction**: sit, touch; 4) **Help**: tutorial, help. 'describe' is one of the most frequently used commands. When typed without any arguments, it lists the names of objects and avatars found around the user's avatar. Users can learn more about an object or an avatar by typing 'describe' followed by the name of the object/avatar of interest.

3 SYNTERELLA

A list of objects and avatars may not be considered as a good description of a 3D environment. However, there are certain reasons behind this choice of strategy. First of all, virtual worlds do not have a predefined story line. Users are free to do anything the world offers at any time. For example, a user may look for some old friends nearby. S/he may choose to take a walk on the beach or go to a bar and be social. Then, s/he may teleport to a random island to discover new places. This means, unlike games, it is not easy to predict what the user is going to do next, based on their current location or past actions. Without a priori knowledge about the intentions of the user, task-based descriptions cannot be generated. Additionally, we can assume that the users benefit from knowing about objects and avatars around them no matter what they are planning to do. Finally, the virtual worlds provide a fair amount of useful textual information with a limited variety and these are mostly related with the objects and the avatars.

Even though we simplify the task to describing objects and avatars in an environment, it is still very challenging to create useful textual feedback. One of the problems is the high population of objects in the virtual environments, just like the real world. Even the smallest room may contain tens of objects. Let alone the details, only the names of objects may be more than enough to overwhelm the user. In some locations, a 'describe objects' command may result in a very long spoken feedback that lists the names of hundreds of objects (Figure 2). On the other hand, users may want to learn more about each object in a less crowded environment without having to type 'describe' again and again (Figure 3). Iterative queries slow down the interaction by requiring more commands to be typed and it also bores the user.

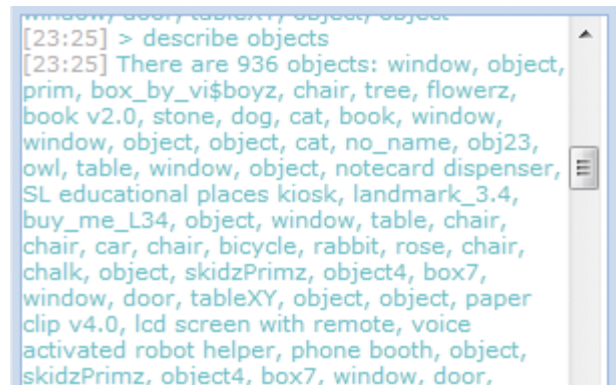


Figure 2. High population of objects may cause extremely long textual feedback.

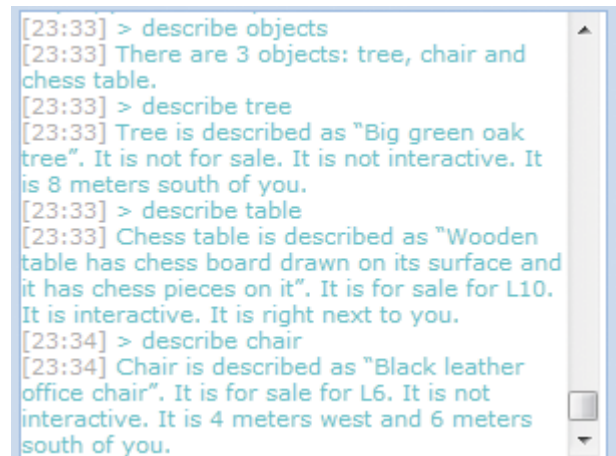


Figure 3. Users iteratively query for more details.

Another problem is that the users may not be interested in objects which are far away from their avatars. However, the definition of being far changes from user to user. Similarly, users may use different speeds in screen readers and respectively they may prefer textual descriptions with different lengths. For example, a user may get overwhelmed by 50 words length of feedback whereas another one may listen to 200 words without a problem. Therefore, a configurable feedback synthesizer should be employed instead of a one-size-fits-all descriptor.

Syntherella is our proposed feedback synthesizer which can dynamically adjust the amount of information based on the object density of an environment and the preferences of the user. The feedback synthesis has five steps:

1. Scanning
2. Filtering
3. Thresholding
4. Grouping
5. a) Aggregation or b) Detailing

Syntherella uses two parameters, namely, preferred word limit (`pref_word_lim`) and preferred range limit (`pref_range_lim`). The default values for these parameters are 100 (words) and 10 (game meters) respectively. The user can change these parameters anytime. `pref_range_lim` is used in the scanning and thresholding steps and `pref_word_lim` is used in the grouping, aggregation and expansion steps.



In the next six subsections, we explain the steps of feedback synthesis. The raw description of a scene shown in Figure 2 is used as the base for the process.

3.1 Scanning

The first step of the feedback synthesis is scanning. In this step, the system scans for the objects around the avatar. Assuming that the user is more interested in nearby objects than the others, the ones located outside of the scanning range are filtered out. In visual viewers, a similar strategy is being used and among all the objects in the avatar's field of view, only the ones located within a certain distance are rendered. An important difference of our approach is that to avoid having the user turning in different directions to find out what is around, all objects within 360° are considered. The radius of the scanning circle is defined by `pref_range_lim`. Figure 4 shows the feedback generated after the scanning step.

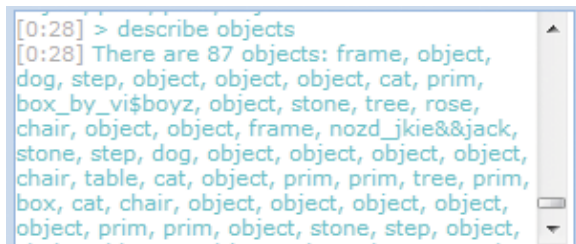


Figure 4. After the scanning step, the feedback includes only the objects located in the scanning circle.

3.2 Filtering

Non-game-like virtual worlds (e.g. Second Life [7], OpenSim [8]) are almost completely created by their communities. Users contribute to virtual worlds by creating objects (from clothes to furniture, from buildings to planes, from trees to islands). However, as the majority of the content creators assume that users can see, they do not worry about providing descriptive names for their objects and leave them with the default value, “object”. However, an object named “object” means nothing to a screen reader user. This is a very serious problem about accessibility in virtual worlds. It is found that 32% of the objects are called “object” in Second Life [3]. Additionally, some users label their objects with gibberish or non-descriptive names like “SkidzPrimz” or “byJohnnyDoe1289”. In a previous work, Juan and Folmer designed a scavenger hunt game called Seek’n Tag [14] where sighted virtual world users are asked to find and label objects. The game is designed based on the games-with-a-purpose (GWAP) paradigm [13]. The purpose of this game is to reduce the number of non-descriptive object names.

Non-descriptive names cause the textual feedback to be longer while not providing any information. Therefore, in the filtering step, the system checks each object’s name with a blacklist and filters out the ones with undesired names. Users can contribute to this list by reporting objects with non-descriptive names. Figure 5 shows the feedback generated after the filtering step.

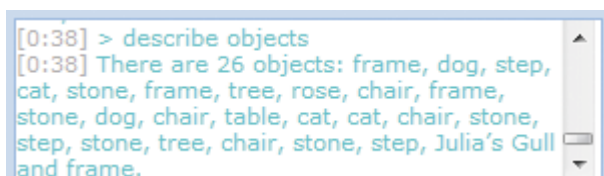


Figure 5. Filtering removes non-descriptive object names.

3.3 Thresholding

Lack of knowledge about the intention of the user limits the system’s ability to optimize the feedback’s effectiveness since it cannot heuristically predict which pieces of information are more important for the user. This limitation led us to create a more generic approach that fits most of the situations. Interaction is an important aspect of virtual worlds and interactive objects generally attract sighted users’ attention more than static objects. Therefore, we want to facilitate the interaction of screen reader users with the environment by highlighting interactive objects. We also assume that bigger and closer objects are more important than smaller and farther ones. Additionally, we assume that the longer the description of an object gets the more details it provides. The same is also true for its name. Based on these assumptions, the system decides which objects are more important and it applies thresholding to filter out the least important ones.

The system assigns five property importance values to each object for its size (s), distance to the user’s avatar (m), name length (n), description length (d) and interaction capabilities (i). Each object consists of one or more simple components called primitives. Assume that os_i is the volume of a primitive i in the object, s_{max} is the maximum size constant, op is the position vector of the object, ap is the position vector of the avatar, onl is the number of characters in object’s name, nl_{max} is the maximum name length constant, odl is the number of characters in object’s description, dl_{max} is the maximum description length constant and ois is the scripted constant (< 0.5). These constants are used in the following equations to calculate the property importance values:

$$s = \min(1.0, (\sum os_i) / s_{max}) \quad (1)$$

$$m = 1.0 - (|op - ap| / pref_range_lim) \quad (2)$$

$$n = \min(1.0, onl / nl_{max}) \quad (3)$$

$$d = \min(1.0, odl / dl_{max}) \quad (4)$$

$$i = \begin{cases} ois & \text{if scripted} \\ 1.0 & \text{if touch scripted} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Then, an overall importance value (I) is calculated as follows:

$$I = s*sw + m*mw + n*nw + d*dw + i*iw \quad (6)$$

$$sw + mw + nw + dw + iw = 100 \quad (7)$$

where sw , mw , nw , dw and iw are importance weight constants for size, distance, name length, description length and interaction capabilities respectively. Objects with an importance value lower than the predefined threshold are removed from the objects list. This is similar to Furnes’ use of the degree of interest (DOI) function and a threshold value in fisheye views to obtain the most interesting subset of items in [4].

This approach helps eliminate the objects of less importance from the description. For example, a small stone at the edge of the scanning range with no interaction capabilities cannot make it to the next step. Obviously, thresholding causes information loss but the space opened by the removal of these relatively unimportant objects may let Syntherella give more details about probably more important objects or at least can prevent it from losing more valuable information in the next steps. Figure 6 shows the feedback generated after the thresholding step.

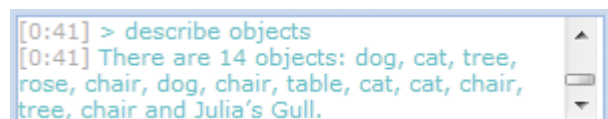


Figure 6. Thresholding removes objects of little or no importance.



3.4 Grouping

It is quite usual to see more than one instance of the same object in an environment. For example, there are many trees of the same kind in a forest or there are many identical chairs, tables or computers in an office. As virtual environments are generally imitations of the real world, the same is also true for these environments. Therefore, grouping the objects having the same name helps having a shorter, simpler and more fluent feedback. Figure 7 shows the feedback generated after the grouping step.

```
[0:44] > describe objects
[0:44] There are 14 objects: 2 dogs, 3 cats, 2
trees, rose, 4 chairs, table and Julia's Gull.
```

Figure 7. Grouping creates a shorter and more fluent feedback.

Grouping is the last step before the generated feedback's word count (`word_cnt`) is compared to `pref_word_lim`. The comparison process is not strict like thresholding. Instead, it employs a tolerance factor (`tf`). There are three ways that feedback can be handled after this point:

Leave the feedback as it is if

$$word_cnt \in (1-tf) * pref_word_lim, (1+tf) * pref_word_lim$$

Apply aggregation if

$$word_cnt > (1+tf) * pref_word_lim$$

Apply detailing if

$$word_cnt < (1-tf) * pref_word_lim$$

3.5 Aggregation

Classification of real world objects is a very challenging task because of their variety, different perceptions of individuals and partial similarities and differences between objects. Classification of virtual world objects is even more challenging because creators are not limited to the rules of physical world which lets them create objects that may not be fit in any object class previously defined. On the other hand, even a simple taxonomy of virtual world objects can be of great use for our purposes. This taxonomy can help in grouping related (i.e., not necessarily identical) objects together which can result in more concise feedback as requested.

We used an automated avatar to collect virtual world objects from a number of different environments. This avatar reported the objects detected at each new location it teleported to. Using these samples, we manually created a number of general classes that can cover the majority of the objects. Seven object classes are created under the root class ("known objects"). These are "plant", "animal", "tool", "structure", "vehicle", "furniture" and "apparel". Each of these classes has a number of subclasses which result in a multi level taxonomy. One problem worth mentioning here is the variety in names given to the same object type. For example, three cats may be named as "katto", "cat v2.4 by Hans Hjuri" and "kitty kitty". This means even if the taxonomy has a rule saying "a cat is an animal", Syntherella still needs to identify a cat from its name first. Otherwise, it is labeled as an "unknown object". However, manually adding a new rule to the taxonomy for every single object is not feasible. The object labelling game Seek'n Tag, mentioned in Section 3.2, helps improving the virtual world taxonomy by creating rules for labelled objects. Despite these efforts, the number of non-descriptive named objects is increasing as the virtual world users do not feel the need to correctly name their objects. Therefore, in another on-going work, we are using computer vision techniques to be able to classify the virtual world objects without using their name attributes.

In this step, the taxonomy mentioned above is used to group objects together and consequently decrease the number of words used in the generated feedback. The unknown objects (i.e., the ones which are not present in the taxonomy) are also grouped together. The aggregation step starts by separating known and unknown objects into two groups. Then, a new feedback is generated which only includes the names and the number of members of each group. If `word_cnt` is below the desired range of `pref_word_lim`, the process is repeated by going one level down in the taxonomy (creating less general classifications). This loop continues until having a word count in the desired range of `pref_word_lim` or until reaching the leaf classes in the taxonomy. Note that the objects classified as "unknown" are not further processed after the first round. This aggregation approach not only generates a feedback with the desired length but also keeps the feedback consistent by using the same level of generalization for all objects. To be able to generate a feedback, it is required to set the `pref_word_lim` to a value above 7 because at the root level, the longest possible feedback has a word count of 8 (e.g. "There are 7 known and 2 unknown objects."). In the case that there is only one object found in a class and the object name is not significantly longer than its class name, aggregation is not applied to that object and instead of its class name, its actual name is included in the feedback. This approach allows providing more specific information with the same feedback length. Figures 8 and 9 show the feedback generated after the aggregation step.

```
[0:53] > describe objects
[0:53] There are 3 plants, 5 animals, 5
furniture and Julia's Gull.
```

Figure 8. Feedback after aggregation with a `pref_word_lim` of 14. Since Julia's Gull is the only object in the unknown objects class, its name is used in the feedback instead of its class name.

```
[0:54] > describe objects
[0:54] There are 13 known and 1 unknown
objects.
```

Figure 9. Feedback after aggregation with a `pref_word_lim` of 8.

3.6 Detailing

When `word_cnt` is below the desired range of `pref_word_lim`, detailing is applied. Until this step, the generated feedback consisted of only the names or classes of the objects along with their counts. Now, since it is known that there is available space for more information, Syntherella can provide more details about each object. However, it is necessary to provide the same amount of details for all objects not to confuse the user with inconsistent feedback.

There are only a limited number of information types that is provided by the virtual world. Some of these are the object's name, description, location, size, color, sound, price tag and interaction capabilities. Additionally, the lack of knowledge about the intentions of the users prevents us from selecting the most useful information from the set of available ones. Therefore, we decided to create a standard ordering of the available information that can be the most useful on average situations.

Spatial information about the objects is almost always useful for navigation and interaction of the user's avatar. The shortest way of providing this information is to use four main directions (west, north, east and south). Among the other available pieces of information, object description is usually the most informative one since it may contain info about the material the object is made



of, the usage of the object, the permissions related with the object (such as copying and editing) or other details about the content. Interactive objects and the objects on sale frequently attract the attention of the users. Therefore, the ordering is defined as: name, direction, description, interaction capabilities, sale price.

It should be noted that fixed ordering strategy may not work well in some situations. As future work reveals the methods on learning the intentions of the users, the most useful pieces of information will be selected separately for each new situation.

Even though we have an ordering in hand, without an efficient way of feedback generation, the resulting description will not be as effective as desired. This is because some of the objects have properties in common such as interaction capabilities or direction; and sequentially listing all objects along with their properties would cause repetitions in the generated feedback which would consequently result in confused and overwhelmed users. Additionally, listing all the properties of an object next to its name creates a feedback which is very hard to follow. A sample feedback that illustrates these problems is given below:

There is car (it is described as “yellow sports car with two doors”; it is interactive and it is for sale for L200; it is 20 m west and 6 m north of you), window (it is described as “big triangle shaped window”; it is not interactive but it is for sale for L19; it is 12 m east of you), window (it is described as “big triangle shaped window”; it is not interactive but it is for sale for L19; it is 12 m east and 1 m south of you), window (it is described as “big triangle shaped window”; it is not interactive but it is for sale for L19; it is 12 m east and 2 m south of you), car (it is described as “yellow sports car with two doors”; it is interactive and it is for sale for L205; it is 10 m west and 6 m north of you).

To remove confusion, we divide the feedback into two parts where the first part contains only a list of objects, grouped by their directions and the second part provides details about each object. This way, the user will have a priori knowledge on which objects s/he will be informed about. This approach is similar to going over news headlines before providing details for each story. In the first part, Syntherella uses four main directions to group objects only once whereas in the second part, it incrementally uses all of the remaining features to create different groupings. For detailing, a 6-level tree structure is used where each level represents a different type of information. The process starts by generating the first part of the feedback without using the information tree. The second part of the process begins by creating groups using only the first and second levels of the information tree. The number of members for each group is defined by the number of leaf nodes connected to the parent node located on the second level. Using the member counts and information obtained from the first two levels, the second part of the feedback is generated. If word_cnt is below the desired range of pref_word_lim, the second part of the process is repeated by including the next level in the tree. This loop continues until having a word count in the desired range of pref_word_lim or until reaching level 6. This approach keeps the generated feedback consistent at each level while minimizing repetitions in the feedback. For an illustration, Figure 10 shows the information tree for an environment with three cats and two chairs. The first part of the feedback generated for this environment is shown below:

There are 2 cats to your north; 1 cat to your south and 2 chairs to your east.

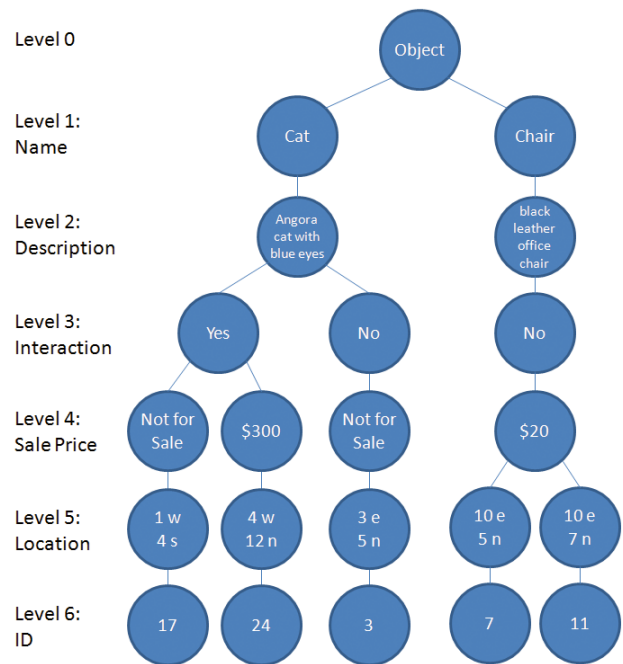


Figure 10. A sample information tree for 3 cats and 2 chairs.

Four main directions are used in the first part to keep the feedback shorter and simpler. Note that the object locations are approximated to fit in one of these four groups. The second part of the feedback at each level is shown below:

Level 2: 3 cats are described as “Angora cat with blue eyes”. 2 chairs are described as “black leather office chair”.
Level 3: 3 cats are described as “Angora cat with blue eyes”. 2 of them are interactive. 1 of them is not interactive. 2 chairs are described as “black leather office chair”. They are not interactive.
Level 4: 3 cats are described as “Angora cat with blue eyes”. 1 of them is interactive but not for sale. 1 of them is interactive and for sale for L300. 1 of them is not interactive and not for sale. 2 chairs are described as “black leather office chair”. They are not interactive but for sale for L20.
Level 5: 3 cats are described as “Angora cat with blue eyes”. 1 of them is interactive but not for sale. It is 1 m west and 4 m south of you. 1 of them is interactive and for sale for L300. It is 4 m west and 12 m north of you. 1 of them is not interactive and not for sale. It is 3 m east and 5 m north of you. 2 chairs are described as “black leather office chair”. They are not interactive but for sale for L20. 1 of them is 10 m east and 5 m north of you. 1 of them is 10 m east and 7 m north of you.

The location property is placed in level 5 since it is the most differentiating one among all. In other words, it causes more groups to be created which in turn results in longer feedback. Note the difference in word_cnt between the feedback generated at levels 4 and 5. Besides, location information may not be needed by the user since the first part of the feedback has already provided basic spatial information (i.e., directions). Level 6 holds the unique id property for the objects which ensures that there is only one object at each leaf node. This is useful for the rare situations where the objects are built on the top of each other (i.e., same 2D coordinates).

Figure 11 shows the feedback generated after the detailing step.



```
[1:08] > describe objects
[1:08] There are 2 dogs, 1 cat, 2 trees to
your north; 4 chairs, 1 table, 1 rose to your
west and 2 cats, Julia's Gull to your south. 2
dogs are described as "black and brown Saint
Bernard". 3 cats are described as "Angora cat
with blue eyes". 1 tree is described as "large
oak tree". 1 tree is described as "green and
tall palmae tree". 4 chairs are described as
"brown stylish dining chair". Rose is described
as "Red rose". Table is described as "diner
table for 4". Julia's Gull is described as
"Laughing gull that can fly; designed by Julia
```

Figure 11. Expanded feedback after detailing.

4 DISCUSSION

There are two main types of limitations to our approach. The first one originates from the information source which, in our case, is the virtual world itself. Unless we have a graphical interpreter, our work relies on the amount, variety and the quality of the textual information provided by the virtual world users. As mentioned before, the lack of descriptive names causes our system to neglect some of the objects even though they might be very important for the user. Additionally, the limited variety of information available in text format causes narrow descriptions. Limitations in generalizability seem to be the biggest problem among all. There are still only a few open-source virtual worlds which allow developers to modify interfaces or directly communicate with the servers for useful information. The second type of limitation is about one's perception and some other's description. For sighted users, observing a virtual world using a visual interface is like standing on the doorsill of a very big room and looking inside whereas for the screen reader users, it is more like asking questions to that person on the doorsill about the room. Sighted users can consume all the visual information in a few seconds without overtiming their minds which means they do not need to explicitly state intentions to narrow down the search results. On the other hand, for users with VI, finding what is of interest requires much more effort. Processing every piece of information available cannot be accomplished in the blink of an eye since they can only be served sequentially. Additionally, it is very easy to get overwhelmed by this great amount of information. Knowledge about intentions would surely help in providing the most relevant information but unfortunately, even this may not be enough to fill the gap between perception and description.

5 FUTURE WORK

We hope to see that Syntherella provides more efficient interaction and less cognitive load for the intended audience in practice. User studies with screen reader users will help us evaluate our approach with respect to effectiveness. Therefore, we will focus on designing and conducting user studies.

Syntherella is implemented and integrated to our web-based TextSL client. Descriptor is also available as an alternative feedback generator. We will start collecting data about how the users perform with both, for a wide-range evaluation. Besides, users can always send us their suggestions and comments directly from TextSL interface. Based on these user logs and suggestions, Syntherella will be improved to be able to fulfil the users' needs.

The presented approach shouldn't be limited to virtual worlds only. Users with VI can benefit from descriptions of real world locations and paths. With the help of computer vision techniques, pictures, videos and maps of environments can be transformed into effective spoken descriptions. This approach can, at the very least, facilitate safe navigation for users with VI.

As emphasized before, the most effective feedback can be generated only when the user's goals, interests, purposes or intentions are known at a time. However, instead of focusing on specific needs for each user, more general task categories (e.g. navigation, shopping) can be introduced to the system. Then, user can select a category that fits her current task and based on the selected category, a more relevant feedback can be generated. For instance, even simply changing the order of the information presented in a feedback may improve the performance. Provided that the system has an idea about the user's task, dynamic ordering of the information tree nodes can help to put most relevant information at the beginning of a feedback. The grading function in the thresholding step can also benefit from the usage of heuristics and varying coefficients defined by the task. Importance values assigned to the objects can be biased based on the user's interaction history with similar objects. Additionally, at the same step, object names can be cross checked with a dictionary for better identification.

6 CONCLUSION

Syntherella is a feedback synthesizer that aims to generate effective textual feedback for any preferred length. It converts a list of objects and properties into structured and useful descriptions. The feedback synthesis has five steps: scanning, filtering, thresholding, grouping and aggregation/detailing. The proposed approach can greatly benefit from knowledge about user's intentions. Additionally, computer vision techniques can eliminate the dependency on the incomplete textual data obtained from virtual world servers. This would also allow us to use Syntherella in mixed reality environments.

REFERENCES

- [1] P. Abrahams. Second life class action, http://www.it-analysis.com/blogs/Abrahams_Accessibility/2006/11/second_life_class_action.html. 2006.
- [2] W. S. Carter. Enabling the blind in virtual worlds. In W4A'10: Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility, 2010.
- [3] E. Folmer, B. Yuan, D. Carr, M. Sapre. Textsl: A Command-Based Virtual World Interface for the Visually Impaired. Proc. ACM ASSETS '09, pages 59-66, 2009.
- [4] G. W. Furnas, Generalized fisheye views, Proc. SIGCHI conference on Human factors in computing systems, pages 16-23, 1986.
- [5] IBM human ability and accessibility center. Virtual worlds user interface for the blind, <http://services.alphaworks.ibm.com/virtualworlds/>.
- [6] LibOpenMetaverse, <http://www.openmetaverse.org/projects/libopenmetaverse>.
- [7] Linden Research. Second Life, <http://www.secondlife.com/>.
- [8] OpenSim, http://opensimulator.org/wiki/Main_Page/.
- [9] M. Pascale, S. Mulatto, and D. Prattichizzo. Bringing haptics to second life for visually impaired people. In EuroHaptics '08: Proceedings of the 6th international conference on Haptics, pages 896-905, Berlin, Heidelberg, 2008. Springer-Verlag.
- [10] TextSL, <http://textsl.org/>.
- [11] S. Trewin, V. Hanson, M. Laff, and A. Cavender. Powerup: An accessible virtual world. In Assets '08: Proc. of the 9th international ACM SIGACCESS conference on Computers and accessibility, pages 171-178. 2008.
- [12] Virtual Guidedog Project, <http://virtualguidedog.org>.
- [13] L. von Ahn and L. Dabbish. Designing games with a purpose. Commun. 51(8):58-67, 2008. ACM.
- [14] B. Yuan, M. Sapre, E. Folmer. Seek-n-Tag: A Game for Labeling and Classifying Virtual World Objects. In GI '10: Proc. of Graphics Interface conference, pages 201-208, 2010.

