Virtual World Accessibility with the Perspective Viewer

Rynhardt Kruger and Lynette van Zijl

[1]Stellenbosch University, Department of Computer Science,
Stellenbosch, South Africa
rpkruger@ml.sun.ac.za, lvzijl@cs.sun.ac.za

ABSTRACT

Virtual worlds are mostly inaccessible to blind users, due to their exclusively graphical interfaces. We developed the Perspective viewer for Second Life and OpenSim virtual worlds, which enables blind users to explore and navigate Second Life virtual worlds with relative ease. The Perspective viewer allows different tools to be modularly added to the viewer, to enable comparison amongst different tools proposed in accessibility research. The Perspective viewer and its set of navigation and exploration tools are described here, with an analysis of the types of tools essential to the blind user in the navigation and exploration of virtual worlds. In particular, the manner in which the audio channel uses and filters information in the navigation of the virtual world, is highlighted. The development of the viewer enabled us to make several recommendations for virtual world accessibility standards.

1. INTRODUCTION

Virtual worlds are virtual representations of physical environments. These virtual worlds are typically hosted online, and can be accessed through a virtual world viewer, analogous to a web browser. The user creates an avatar as a digital representation of himself, in order to explore the virtual world. The avatar therefore acts, in a certain sense, like a cursor in an electronic document – it establishes the position of the user in the virtual world.

Virtual worlds usually provide output in a primarily graphical fashion, to allow (sighted) users to interact with the virtual world environment. Audio is used as a secondary output medium, but usually only for a more realistic simulation of the virtual world rather than to convey essential navigational information. Graphical output is inaccessible to blind users, since most screen readers cannot interpret graphical content. Therefore, an alternative medium must be used to provide access to virtual worlds for blind users. To this end, a virtual world client (including a viewer) was developed, which provides interaction with the virtual world through audio and synthesized speech.

Our virtual world client, called Perspective [1,2], has a three-fold use. Firstly, it can be employed by blind users for accessing virtual worlds. Secondly, it can be utilized to study accessible methods for representing the state of a virtual world. And thirdly, it can be used to evaluate virtual world accessibility in general, in order to work towards an accessibility standard for virtual worlds, similar to the Web Content Accessibility Guidelines [3].

In this paper we present the Perspective viewer (see Figure 1) and the navigation and exploration tools it provides. Experience with the viewer leads to our findings on the state of virtual world accessibility, specifically with regards to Second Life and compatible virtual worlds. In particular, virtual world building practices which will increase accessibility, are noted. Also, it is shown that an accessibility paradigm, similar to graphical user interface accessibility paradigms, can be used to extend the information provided to virtual world clients, to present a more accurate representation to the blind user.

Figure 1. Virtual world viewer in use at an OpenSim virtual conference [2].

2. BACKGROUND

Accessibility for the blind in virtual world environments has received some attention in the literature. White et al [4] analyzed accessibility issues, and made recommendations for the improvement of accessibility in virtual worlds. Folmer et al [5] developed the TextSL viewer for Second Life, based exclusively on textual descriptions. The user controls their avatar by means of textual commands, and all objects in the virtual world are presented by textual descriptions. The use of audemes (short duration sounds) to identify concepts was proposed by Ferati et al [6] in an information reviewing tool for blind children. Audemes are applicable in virtual worlds as well. More recently, Maidenbaum et al [7] proposed the use of Sensory Substitution Devices which provide visual information in virtual worlds via the other senses. For example, they developed the Eyecane, which can be used for object recognition and navigation. In addition, certain audio games developed specifically for accessibility, can also contribute to virtual world

accessibility techniques. Examples include PowerUp by Trewin et al [8], Terraformers by Westin [9], AudioQuake [10] and Shades of Doom [11]. Lastly, the accessiblity techniques for standard graphical user interfaces may also be useful in the case of virtual worlds. In particular, the question of focus played a notable part in the navigation tools in the Perspective viewer.

The reader may note that in the systems above, tools were developed that were specific to the given implementation or game. In contrast, the Perspective viewer allows for the modular interchangeability of tools, thus providing a research environment with which such tools may be compared and evaluated.

Essential to the development of an accessible virtual world client, is the protocol of the virtual world itself, as the protocol determines what information about the virtual world to expose to the client. As Second Life [12] is currently the most widely used virtual world, the Perspective client was developed for Second Life (and the compatible opensource virtual world OpenSim [13]). The information typically available to Second Life virtual world clients, and how they can be used by an accessible client, are described below.

2.1 Virtual Worlds

Virtual world software consists of two parts: a server maintains the model of the world being emulated, as well as the current state of the world. Client programs connect to the server and present the world to the users. The communication between server and clients is done by the use of a common protocol.

In the Second Life protocol, the state of the virtual world is represented by the state of each object in the region where the user's avatar is currently located. An object can consist of one or more primitives. Each primitive describes a shape, such as a cylinder or a cube, and attributes such as the scale, rotation, and current position of the primitive. Each primitive also contains a list of properties, containing extra information about the primitive. Common properties include the primitive's name, description, access permissions, and so on.

The current Second Life protocol enables alternative representations of the virtual world through the attributes and the properties of each primitive. Object names, along with their positions, can be conveyed to the blind user via 3D audio and synthesized speech. However, with the current protocol, the clients are unable to identify the type of an object, for instance whether it is a chair or a door. Blind users typically require more state information as well, such as whether a door is open or closed, or whether a switch is enabled or disabled. For such functionality to be available, a paradigm similar to graphical user interface accessibility should be used.

2.2 Graphical User Interface Accessibility

Accessibility APIs are an established method of providing accessibility for graphical user interfaces. These interfaces may be desktop environments, but more recently also mobile platforms. Common accessibility APIs include the Android Accessibility API [14], the Accessible Service Provider Interface (ATSPI) on Linux and Unix desktops [15], and the Microsoft Active Accessibility framework on Windows [16]. Most current accessibility APIs were developed for 2D graphical environments. However, the methodology employed can be adapted for 3D environments such as virtual worlds.

An accessibility API serves as a bridge between the graphical user interface (an application or the desktop environment itself), and a screen reader. The accessibility API converts each component of the user interface into an object that can be queried by the screen reader. The accessibility API also notifies the screen reader when a user interface event occurs, for example when a component gains focus or a key is pressed. The screen reader interprets the information made available by the accessibility API, and presents it to the user via synthesized speech and/or Braille.

Each object maintained by the accessibility API contains three aspects that can be queried by the screen reader: roles, states, and properties. A role denotes the type of underlying user interface component, and the actions possible with the component. A role may for instance indicate that the component is a pushbutton, and thus the user can click it. A state indicates the current state of the component; for example, whether a checkbox is checked or unchecked, or a pushbutton is pressed or not. A property contains extra information about the component, such as its position on the screen, its name and its description.

All the objects maintained by the accessibility API are collected in a tree structure, denoting their logical relation to each other. The root of the tree may for instance be the toplevel window of the current application. Its direct children will then be the menubar of the application, and the component denoting the content area of the application. The content area itself may be a complex component such as a treeview or table, in which case it will have children itself.

Two standards are maintained by the World Wide Web Consortium, to ensure accessibility on the web. These standards serve as guidelines for web developers, as well as providing HTML tags for converting information on the web to a format suitable for an accessibility API. The Web Content Accessibility Guidelines (WCAG) [3] is a set of guidelines for web developers. Guidelines include the use of alternative text for images, and captions for videos. The WCAG also advises on practices to ensure a more accessible experience, such as a link at the top of a page to move to the content. The Accessible Rich Internet Applications (WAI-ARIA) [17], on the other hand, is both a set of guidelines, and HTML tags which can be used to convey information about the properties of a web page to the accessibility API running on the user's desktop. These tags allow a web developer to specify roles for custom components defined within the page. Adding defined roles to components allow screen readers to better interpret the behavior of the components.

The design of the Perspective client draws from most of the principles discussed above, and is discussed in the next section.

3. THE PERSPECTIVE CLIENT

The Perspective client consists of a framework containing a core API, and navigation and exploration tools developed using the API (see Figure 2). The core of the system lies in the bottom two blocks, namely, the specific exploration and navigation tool modules, and the command tool to control the Perspective modules. The terminal provides a dedicated text communication between the user and the viewer, and calls the text to speech libraries. Finally, on the top level, the user interface allows for limited graphical output for sighted users, and text interaction with blind users. The connection to the virtual world is established via the user interface.

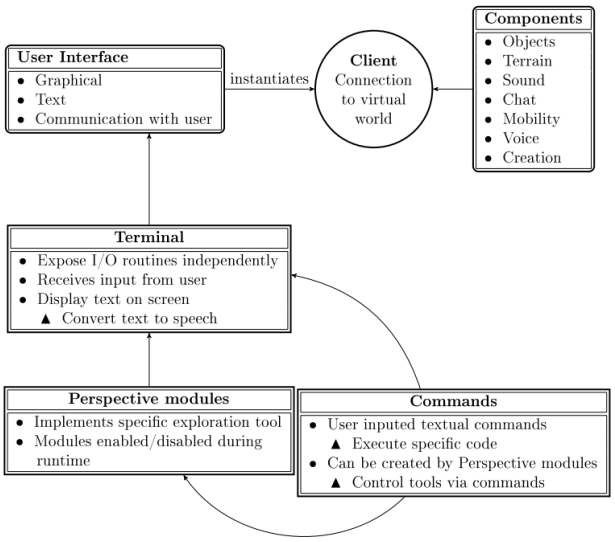

Figure 2. Components of the Perspective viewer

The blind user of a virtual world must be able to navigate (move through) the virtual world, and explore a certain region, all without the benefit of sight. Hence, the objects in the world must be described via the audio channel.[1] This leads to an overload on the audio channel, and our tools were designed precisely to balance the demands on the audio channel, whilst providing the necessary information relevant to the current environment of the user.

The tools allow the blind user to explore and navigate through the virtual environment by several methods. The tools include a virtual sonar, a realtime auditory display of objects around the user's avatar, and a tool allowing exploration of the virtual region as a 3D grid. All the tools make use of the Perspective core API, which in turn makes use of information provided through the Second Life protocol.

3.1 Interaction, focus, object names, and scope

To use the Perspective viewer[1], the user types in textual commands (preceeded by a "/" character), or uses the two types of keystrokes available. The keystrokes are the reading keys, and the action keys. Reading keys allow a user to request that previously spoken output be repeated, and the unit of spoken output may be a line, word or character. Action keys, on the other hand, let the user take actions, such as to control the avatar, or to investigate menu options.

Sighted users typically use a mouse click to focus on a specific object, or click on a menu item to put focus on the menu. In the text-based environment of the Perspective viewer, the functionality of a current focus or focus change is achieved by maintaining a reference to the object currently in focus. Any actions are then performed on the object in focus. To change the focus to a different object, the user has several possibilities. The auto focus tool (see Section 3.2 below) automatically puts the focus on the object nearest to the user's avatar. Alternatively, the user may select the focused object from a menu, or search for an object to receive focus, with (part of) a text string in its name.

To identify a (focused) object to the user, the Perspective viewer identifies the object by its name. This presents a problem if the creator of the object did not give it a descriptive name, which is often the case. Hence, the Perspective viewer maintains two lists of objects that can be queried by the user, namely, a list of named objects and a list of unnamed objects. To improve navigation and exploration, the list of named objects is filtered to remove occluded objects, and remove objects with duplicate names except the one nearest to the user. The removal of occluded objects prevents the user from incorrectly trying to interact with objects not available for interaction, such as an object in another room. The filtering of duplicate objects assists the user in interacting only with the most relevant of the duplicate objects, such as in the case of a room with four walls – the user typically wants to interact only with the wall nearest to the avatar.

Note that it is also necessary to sensibly define user interaction with complex objects, which are constructed from basic objects. Second Life organises complex objects into a so-called link set, with a root object and children objects[12]. The question is whether to identify only the root object to the user, or the children objects as well. Presenting too many objects overloads the audio channel, but if only root objects are presented, important children objects (such as the on/off switch on a machine) may be missed. The Perspective viewer attempts a compromise in this regard. It presents all named objects to the user by default, but the user can filter the objects (using the left scan and right scan tools) to only discover root objects.

The number of objects presented to the user must be manageable over the audio channel. Hence, the Perspective viewer differentiates between objects in local scope, and objects in global scope. Local scope is the area within a 10m radius from the user's avatar, while global scope is the whole of the current region – local scope was chosen to coincide with the chat range in Second Life. The reader may note that most audio games operate in local scope only, as the exploration of the world is part of the game[11].

[1] The use of haptic feedback as an output channel is not considered in this work.

3.2 Navigation and Exploration Tools in Perspective

All navigation and exploration tools in the Perspective viewer either provide continuous output, or provide output only on activation. In addition, user feedback indicated that volume control is essential for continuous output tools, and this feature is available where applicable.

The audio compass tool allows for avatar orientation. The user may request to hear the current direction that the avatar is facing (say, north-north east), or turn their avatar clockwise or anticlockwise, and the new direction is spoken. A similar tool was used in Terraformers [9]. The ability to turn is implemented in the turning keys tool, and is similar in operation to that found in PowerUp [8]. In addition, by pressing the turning keys repeatedly in quick succession, the user can identify objects in the immediate vicinity, as the nearest object to the avatar is identified.

Object identification in the immediate vicinity is also possible with the virtual sonar tool (similar to that in [6]). Here, all the objects that are in the field of vision of the user's avatar, is gathered into a collection. A beep sound in 3D audio is then emitted at the position of each of the objects in the collection, and the names of the objects are spoken.

To identify objects while the avatar is exploring a region, the auto focus tool can be used. This tool computes the nearest object to the avatar, every time that the avatar moves. Focus is then set to that nearest object. This is a useful tool when an area is first explored, but may interfere with other focus-setting tools.

During exploration, the blind user may not necessarily know if their avatar has moved when a movement command is issued (for example, the avatar may be blocked by an object that prevents it from moving). The footstep module plays a footstep sound whenever movement occurs. Note that the Second Life protocol does not support event notification on movement, and the footstep sound tool therefore uses current and previous location to calculate whether movement has occurred.

For quick and effective navigation, particularly around objects, the directional look commands can identify the object to the left, right, and directly in front of the user. As a further aid to navigation, the directional noise functionality allows the user to switch on a tool which continuously plays a different narrowband noise for each compass direction where no objects occur. A similar concept was used in the Shades of Doom game [11].

Additional environmental information during movement is made available by the echo tool. This tool bounces a clicking sound off objects in the immediate vicinity of the user's avatar. Likewise, the material sound tool is meant to convey the type of each object in the immediate vicinity – it plays the default collision sound of the material covering the object.

During exploration, the user may wish to search for a specific object. The Perspective viewer provides a search function for local scope, as well as one for global scope. The user searches by typing in a text string with part of the name of the object. The search results are presented in a menu. The user may also simply query the object menus for local or global scope, or the avatar menu to find a specific object or avatar.

The Perspective viewer implements two important novel tools: the positional speech tool, and the grid explorer. The positional speech tool creates a speech sample of each of the objects in local scope, and places the sample in 3D audio[2] at the position of that object. These samples are played in sequence, but the time duration for each sample is indirectly related to its distance from the user's avatar. Hence, objects that are nearest to the user's avatar can be heard clearly, while objects that are far away are perceived as a cluster of voices. To avoid confusion in a highly populated immediate vicinity (such as a virtual shopping mall), the user may set the number of objects to be announced with the positional speech module.

The grid explorer mimics a blind person's experience in learning the layout of a new environment. When the grid explorer opens, the region is divided into eight three-dimensional blocks. As the user moves around this structure using the arrow keys, a summary of the objects in each block is announced. The summary consists of the names of the objects if there are three or less objects. Otherwise, the name of the first object is announced, followed by the number of remaining objects. The user can increase or decrease the number of blocks in the grid, effectively changing the resolution of the exploration.

The development of the Perspective viewer, and specifically the core API, allowed for the evaluation of the current state of accessibility of Second Life-based virtual worlds. In particular, the information currently accessible through the Second Life protocol was evaluated. It is our view that information similar to the roles, states, and properties provided by current desktop accessibility APIs can increase the accessibility of Second Life and compatible virtual worlds. The reader may note that the development of the tools were essential to lay the foundation for the accessibility recommendations, as it exposed the current shortcomings in the Second Life protocol and the virtual world building practices.

3.3 Navigation and Exploration Example in Perspective

The advantages of the Perspective viewer can be best explained via an example. Consider the room depicted in Figure 3. This is the view that a sighted user has of the room, for navigation and exploration purposes. Note that the room is (probably) rectangular in shape, with one door leading out from the room (behind the viewpoint of the sighted user), and the room is furnished with a desk, two chairs, a book on the desk, and a filing cabinet.

With the usual visual format as seen by the sighted user, the identity of each object is obtained by its shape. The user can interact with each object via mouse clicks. For example, if the user clicks on the chair, this will make their avatar sit on the chair.

[2] 3D audio creates a sound effect from two speakers, and seems to place sound positionally from any direction (above, below, behind, to the side).

With the Perspective viewer and the current Second Life protocol, the blind user will hear the name of each object spoken. The user may for instance hear the word "desk chair", followed by "desk", "book", and "filing cabinet". This assumes that the objects were labeled properly. The user may, in the case of badly labeled objects, instead hear "cabinet" for the filing cabinet, or "Two Cities" for the book (representing the title).

Figure 3. An OpenSim room with desk, chairs, and filing cabinet. Taken from [18].

If improvements to the Second Life protocol is made, and the roles, states, and properties of the room in question is properly defined, the user may hear "cabinet – container" for the filing cabinet, or "desk chair – chair, approach from left" for the chair. The book may be identified as "Two Cities – book, with title A Tale of Two Cities", and the door as "oak door – door, currently closed".

The reader may note the recommended navigation information (approach from left), or the state of objects (door, currently closed). In the next section, we consider these types of recommendations in more detail.

4. ACCESSIBILITY RECOMMENDATIONS

An accessibility standard for virtual worlds should consist of two parts: firstly, a set of guidelines for virtual world builders, and secondly an extension to the virtual world protocol. The guidelines should ensure that relevant information is added to all virtual world objects, to aid in the identification and categorization of objects by accessible clients. The extended protocol should facilitate the transfer of accessibility-related information from the server to the clients.

4.1 Guidelines for Builders

Objects in Second Life compatible virtual worlds can be identified by several means. Each object has a unique UUID (universal unique identifier), which is suitable for programmatic identification of objects, but not for identification by users. For users it is more suitable to use textual labels for identification, and each Second Life object has a name property that can be used to this end. However, object names are not always filled in, resulting in the object having a default name such as "object" or "primitive". Therefore we propose that object names be filled in at all times. This practice should be encouraged by the build tools of clients, by not filling in a default name. The builder should also be prompted to enter a suitable name, and the prompt should explain the role of the name as an accessibility aid.

Objects in Second Life also have a description property, which can be used to add more detail about the object. However, this description property is rarely filled in. We propose that the name property be kept as short as is needed to aid identification, but that more detailed information be entered in the description property. This should also be encouraged by a suitable prompt within the build environment. The object description should be updated as the state and/or shape of an object changes.

4.2 Extensions to the Build Tools and Protocol

Roles are used in graphical user interface accessibility APIs to denote the types of underlying components. However, roles can also be useful to identify the types of virtual objects. Within graphical user interfaces, roles include pushbuttons and checkboxes, but within virtual worlds may include objects such as doors and chairs. Identification of object types will enable the client to categorize objects of a certain type. This will enable users to more easily find specific objects, such as a chair at a convention, or a door to exit a room. Object roles will also enable the user to quickly grasp the behavior of an object, and therefore the actions that can be performed with it. The Perspective client can deduce some object roles by examining the default action of an object. For instance, if the action is to sit, the object is probably a chair. However, this method is unreliable, since unrelated objects may have the same default action. We propose that roles be added to virtual world objects as attributes that can be set at build time.

States are used in graphical interface accessibility APIs to denote the current state of a component. However, states can also be used in virtual worlds to denote the state of an object. In virtual worlds, states of objects may include "open" in the case of a door, or "activated" in the case of equipment. The state of an object influences the actions possible with it. Currently it is not possible to deduce the state of an object with the current Second Life protocol, and hence the user needs to test actions on an object to deduce the state himself. An example of this is to walk through a door, and if that movement is allowed, the door is open. Therefore we propose that states be added to virtual world objects, and be communicated with the Second Life protocol.

Properties are used by graphical user interface accessibility APIs to specify more information about an underlying component. Second Life already includes a mechanism for specifying object properties. However, the list of properties could be extended to be more relevant for accessibility. For instance, it is possible to query the

rotation of a virtual object, but this rotation property only specifies the rotation of its shape, not the logical orientation of the object. The logical orientation of an object could be useful to denote the direction in which a door or a staircase should be approached, or the direction faced when sitting on a chair. Therefore we propose that properties similar to the logical orientation property be added to virtual world objects.

Accessibility APIs designed for graphical user interfaces have an event mechanism for notifying assistive technologies about changes in the user interface. Second Life also has an event mechanism, but we propose that accessibility relevant events be added to this mechanism. Specifically, an event should be added to indicate state changes of an object. This may include a door being opened, or another avatar sitting on a chair. An event should be added to denote collisions between objects, as currently only collisions between avatars are indicated.

5. CONCLUSIONS

The Perspective viewer provides a mechanism for blind users to use virtual worlds; it provides a framework to compare different accessibility tools in the virtual world context; and it enabled us to set up a short set of guidelines and recommendations with respect to virtual world accessibility for the blind.

As future work, we intend to conduct a full user case study with a sizable set of blind participants, to evaluate each tool in full. We also intend to explore more tools, in order to refine the proposed accessibility guidelines and recommendations.

Acknowledgments

This work is based on research supported in part by the National Research Foundation of South Africa (grant number 90584). Financial support of the first author by MIH is acknowledged.

REFERENCES

[1] R.P. Kruger, *Virtual world accessibility: a multitool approach*, MSc thesis, Stellenbosch University, 2014.

[2] R.P. Kruger and L. van Zijl, "*Rendering virtual worlds in audio and text*", in *Proc. 6th Int. Workshop on Massive Multi-user Virtual Environments*, Singapore, 2014, pp. 5:1-5:2.

[3] W3C, "Web Content Accessibility Guidelines (WCAG) 2.0," http://www.w3.org/TR/WCAG20/.

[4] G. White, G. Fitzpatrick, and G. McAllister, "Toward accessible 3D virtual environments for the blind and visually impaired," in *Proc. of the 3rd Int. Conf. on DigitalInteractive Media in Entertainment and Arts*, Athens, 2008, pp. 134–141.

[5] E. Folmer, B. Yuan, D. Carr, and M. Sapre, "TextSL: a command-based virtual world interface for the visually impaired," in *Proc. of the 11th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Pittsburgh, 2009, pp. 59–66.

[6] M. Ferati, S. Mannheimer, and D. Bolchini, "Acoustic interaction design through "audemes": experiences with the blind," in *Proc. of the 27th ACM Int. Conf. on Design of Communication*, Bloomington, 2009, pp. 23–28.

[7] S. Maidenbaum, D.R. Chebat, S. Levy-Tzedek, and A. Amedi, "Depth-to-audio sensory substitution for increasing the accessibility of virtual environments," in *Proc. of the 8th Int. Conf. on Human Computer Interaction*, Greece, 2014, pp. 398–406.

[8] S. Trewin, V. Hanson, M. Laff, and A. Cavender, "PowerUp: an accessible virtual world," in *Proc. of the 10th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Halifax, 2008, pp. 177–184.

[9] T. Westin, "Game accessibility case study: Terraformers: a real-time 3D graphic game," in *Proc. of the 5th Int. Conf. on Disability, Virtual Reality and Associated Technologies*, Oxford, 2004, pp. 95–100.

[10] M. Atkinson, S. Gucukoglu, C.H.C. Machin and A.E. Lawrence, "Making the mainstream accessible: redefining the game," in *Proc. of the 2006 ACM SIGGRAPH Symp. on Videogames*, Boston, 2006, pp. 21—28.

[11] GMAGames, "Shades of Doom audio game," http://gmagames.com/sod.shtml.

[12] Linden Labs, "Second Life Official Site," http://www.secondlife.com.

[13] OpenSim contributors, "OpenSim," http://www.opensimulator.org.

[14] Android developers, "Accessibility," http://developer.android.com/guide/topics/ui/accessibility/index.html.

[15] The Linux Foundation, "atk/at-spi," http://www.linuxfoundation.org/collaborate/workgroups/accessibility/atk/at-spi.

[16] Microsoft Corporation, "Microsoft Active Accessibility," http://msdn.microsoft.com/en-us/library/ms971350.aspx.

[17] W3C, "Accessible Rich Internet Applications (WAI-ARIA) 1.0," http://www.w3.org/TR/wai-aria/.

[18] https://marketplace.secondlife.com/p/NEW-1950s-Private-EyeDetectives-Office-with-DeskChairsLampFanPhoneHatStandMore/158773