UNIVERSITY OF NEVADA, RENO

A Framework for Making Virtual Worlds Accessible to the Visually Impaired

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

by

Manjari Sapre

Dr. Eelke Folmer, Thesis Advisor
Department of Computer Science and Engineering

August 2009
We recommend that the thesis prepared under our supervision by

MANJARI SAPRE

entitled

A Framework For Making Virtual Worlds Accessible To The Visually Impaired

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Dr. Eelke Folmer, Advisor

Dr. Sergiu Dascalu, Committee Member

Dr. Mark Wessinger, Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

August, 2009
“I was born intelligent, education ruined me!”
Abstract

Virtual Worlds have virtually exploded in popularity and have experienced significant commercial success. People from all over the globe are experimenting with the various features that virtual worlds provide. It is deemed to be the future of social networking, online advertising, distant education and potentially surfing the web may closely resemble the immersive experience of virtual worlds. But going forward, reckoning its scope, it has brought back the issues of accessibility and needs attention. Individuals with visual impairments usually use screen readers to access web. Virtual Worlds like Second Life lack textual representation and are not accessible to screen reader users. Since the Second Life like worlds are completely owned and created by their users, most of their content lacks meta information. This becomes a huge barrier to a text-extraction based approach of making virtual worlds accessible to the visually impaired who can benefit immensely from this platform.

This thesis presents the design of a framework which can be used for making virtual worlds accessible to the visually impaired individuals. It provides a command based text interface to Second Life which when plugged in to a screen reader, enables the individual with a visual impairment to access Second Life. We then propose a technique which can feed meaningful meta data to the objects in Second Life, which can then be retrieved as text and processed to make sense to the screen reader users.
To my mother and father...
Acknowledgements

It is pleasure to thank all and sundry who have helped and inspired me all through my research study.

I offer my sincerest gratitude to my advisor, Dr. Eelke Folmer, without whose perpetual guidance, support and encouragement, this thesis would not have been possible. I would like to thank him for always being accessible, motivating, and for his efforts and willingness to help me throughout my Masters program with everything.

I am grateful to Dr. Sergiu Dascalu and Dr. Mark Wessinger for being on my committee and spending worthy time to review and advice.

I would like to thank Dr. Bei Yuan for her diligence and contribution in numerous ways towards the research.

I express my gratitude to Dr. Mircea Nicolescu, Dr. Monica Nicolescu, Dr. Harris Frederick and Dr. Murat Yuksel who have been a source of inspiration to me during my course of study. I cherished being a part of the Department of Computer Science and Engineering that provided a great working environment and a wonderful experience.

My deepest gratitude to my husband, Kedar Deshpande, for motivating and being there for me unconditionally; to my parents, parents-in-law, my both sisters, brothers-in-law and family for their unflagging love and support. I would like to add a very special thanks to my friends Hrishikesh Kulkarni, Navid Fallah, Juan Quiroz and Durgesh Rani for providing valuable comments, being there for me all through and making my school life enjoyable and unforgettable.

This research was supported by NSF Grant IIS-0738921.


## Contents

### Abstract

### Acknowledgements

### Table of Contents

### List of Figures

### List of Tables

#### 1 Introduction

- 1.1 Virtual Worlds: Overview ................................................. 1
- 1.2 Motivation ................................................................. 5
- 1.3 Proposed Solution ...................................................... 6
- 1.4 Thesis Sections .......................................................... 9

#### 2 Background and Related Work

- 2.1 A Survey of Accessible Games for the Visually Impaired Individuals ... 10
- 2.2 Virtual World Accessibility ............................................. 14
- 2.3 Image Labeling ............................................................ 17
- 2.4 Virtual World Object Labeling .......................................... 21

#### 3 TextSL

- 3.1 Design Requirements .................................................... 25
  - 3.1.1 Audio Feedback ......................................................... 25
  - 3.1.2 Text-based Interface ................................................ 26
  - 3.1.3 Implementation ...................................................... 27
- 3.2 TextSL Functionality ................................................... 28
  - 3.2.1 Getting Started ...................................................... 28
  - 3.2.2 TextSL: An Interpreter .............................................. 29
  - 3.2.3 TextSL Commands .................................................... 30
  - 3.2.4 Findings ............................................................... 35
- 3.3 TextSL Architecture .................................................. 37
  - 3.3.1 Libsecondlife ........................................................ 39
  - 3.3.2 TextSL Interface ..................................................... 42
  - 3.3.3 TextSL Core .......................................................... 42
  - 3.3.4 TextSL Abstraction Layer ......................................... 43
Contents

3.4 Summary ............................................. 45

4 Seek and Tag ........................................... 46
  4.1 TSL DataServer NPC ................................. 47
    4.1.1 Functionality .................................. 47
    4.1.2 Architecture .................................. 48
    4.1.3 Data Collection Experiment and Results ...... 50
  4.2 Labeling Objects ................................. 50
  4.3 Seek and Tag Game ................................. 53
    4.3.1 GamePlay ...................................... 53
    4.3.2 Design .......................................... 55
      4.3.2.1 Seek and Tag HUD .......................... 56
      4.3.2.2 TSL Agent NPC ............................. 60

5 User Studies ........................................ 63
  5.1 TextSL User Study ................................. 63
    5.1.1 Design .......................................... 64
    5.1.2 Results ........................................ 66
  5.2 Seek and Tag User Study ......................... 70
    5.2.1 Design .......................................... 70
    5.2.2 Results ........................................ 71

6 Discussion and Future Work ......................... 73

7 Conclusion ............................................ 78

8 Glossary ............................................... 80

Bibliography ........................................... 84
List of Figures

1.1 Object Modeling Tool .............................................. 4
1.2 TextSL Approach ..................................................... 6

2.1 Shades Of Doom ....................................................... 11
2.2 Audio Quake .......................................................... 11
2.3 Terraformers .......................................................... 13
2.4 Powerup ............................................................... 13
2.5 IBM’s Web Interface to Second Life ............................... 16
2.6 GWAP ................................................................. 20
2.7 Google Image Labeler ............................................... 20
2.8 Phetch ................................................................. 21
2.9 IBM’s Annotation Web Page ........................................ 22

3.1 Zork ................................................................. 27
3.2 Second Life .......................................................... 37
3.3 TextSL ............................................................... 38
3.4 TextSL Design ........................................................ 39
3.5 TextSL Architecture .................................................. 44

4.1 TSL DataServer NPC Design ....................................... 47
4.2 Automatic Labeling using Object Categorization ............... 51
4.3 The Big Picture ....................................................... 52
4.4 Seek and Tag Snapshot .............................................. 54
4.5 Seek and Tag HUD Master-Slave Architecture .................. 57
4.6 TSL Agent NPC Design .............................................. 61
4.7 Seek and Tag Flow Diagram ....................................... 62

5.1 Average Task Performance Times .................................. 67
5.2 Usability Comparison TextSL vs SL .............................. 69
5.3 Accuracy: Manual Naming vs Seek and Tag .................... 71
List of Tables

3.1 TextSL Commands ................................................. 30
3.2 libSecondLife Classes ............................................ 41
5.1 TextSL User Study ............................................... 65
Chapter 1

Introduction

1.1 Virtual Worlds: Overview

Virtual worlds offer rich three dimensional (3D) vibrant environment for interaction and have grown to be a huge business involving millions of players worldwide [1]. These virtual worlds allow the user to control a digital puppet, called an avatar, with human like capabilities like walking, running, flying, gesturing, communicating, etc. Virtual worlds surpass the two-dimensional (2D) internet by offering a much higher degree of interaction. Social interaction possibilities offered by virtual worlds significantly exceed that based on 2D chat rooms, web forums, or messenger clients, providing an engaging and compelling experience to the user. Virtual world interaction mimics real-life interaction. Users can explore the vast world, communicate to other avatars and objects, and be involved in numerous activities like dancing, building objects, playing games, organizing events, etc. The rich immersive 3D graphics, the large amount of user generated content and social interaction opportunities offered by the greater sophistication of virtual worlds could someday eventually make for a more interactive and
informative world wide web[2].

Second Life [3] and World of Warcraft [4] are two of the most popular virtual worlds [1, 5]. These can be distinguished as coming from two different families of virtual worlds. Game-like virtual worlds such as World of Warcraft or Star War Galaxies are modeled after role playing games with typical elements of most video games such as beating enemies, attaining levels, a storyline, goals to achieve, and the possibility of the avatar to die. The second family is the non-game-like virtual worlds. Second Life, There, and Active Worlds are some examples of this type that lack the game-like elements. One of the key differences between the current game-like and non-game-like virtual worlds is that the non-game-like virtual worlds are completely created and owned by their users. Though non-game-like virtual worlds are modeled after the three dimensional graphics and have basic control features similar to that of game-like virtual worlds, they offer a variety of other features and usages right from playing games, socializing, visiting places in the world, and numerous communities and events. Studies reveal that social interaction is the most important aspect of virtual worlds [6].

Second Life is one of the popular non-game-like virtual world, launched by Linden Lab in 2003, and has grown explosively since then with more than 13 million accounts and around 50,000+ users logging in from all over the world on a weekly basis [1]. Second Life can be considered a social community which enables its users to inhabit and interact via role playing dummies called avatars, and offers a high degree of customizability. Users can explore, meet other residents, socialize, participate in individual and group activities, travel throughout the world, and create and trade content including virtual property and services with one another. The Second Life world is vast and contains millions of user generated content. The user created content can be traded using a virtual currency called Linden Dollars, which can be exchanged for real currency.
The land purchasing and renting also is done using the Linden dollars and the rates vary with time. As a result a thriving economy has emerged where a number of people can make living by doing a virtual business in Second Life [7]. The application used to access the Second Life world is a Viewer which is freely downloadable. The Viewer is similar to a web browser using which specific places/regions can be visited in the virtual world depending on user’s choice, whilst it communicates to the Second Life region server to establish a session and receives the information in real-time to render and display inside the Viewer. The viewer supports a variety of basic controls like navigation of the avatar, communicating to other avatars and objects, exploring the world, creating objects, etc.

All the Second Life content is created and owned by its users. Though Second Life can be accessed freely using the Viewer, users must rent or buy land to be able to create content on that land, such as having a personal house, building, or for other activities. There are various ways in which objects can be created, but most commonly it is done by using the 3D object modeling tool built-in within the Second Life viewer. Objects in Second Life are constructed from prims (primitives) which are simple solid geometrical objects. A prim can be constructed from seven building block types or shapes viz. box, sphere, cylinder, torus, prism, tube, and ring. A prim has several physical properties like size, position, rotation, taper, twist, top shear, hollow, path-cut, phantom, physical (Figure 1.1). Objects that need to be passed through by avatars or objects are marked as “phantom”. A part of the prim can be cut using “path-cut” to make a different shape. The objects that are required to simulate physics(objects that have mass, momentum, gravity, and are subject to basic laws of physics) are marked “physical”. This is useful for creating scripted objects such as vehicles and other objects that need to follow laws of physics. Second Life world uses a physics simulation software called Havok Physics [8] for this purpose. All these prim properties can be tweaked in
Figure 1.1: Object Modeling Tool [3]

combination and permutation in various ways to create the required shape. These prims can further be linked (grouped) and textured to obtain the desired object. Objects can also interact in Second Life which is made possible by attaching scripts to them. The scripts use Linden Scripting Language (LSL) that give behavior to the objects. Second Life also allows for importing objects from external 3D modeling tools for creating more complex objects such as the skin or hair of an avatar. Second Life thus offers rich and sleek features to boost the creative side of the players.
It is recommended to go through Chapter 8 for a glossary of definitions of various terms which are commonly used in the virtual worlds like Second Life and will be used throughout the thesis.

1.2 Motivation

Second Life has gathered attention from universities and developers from all over the world for researching different possibilities [9–11] and is also being used as virtual classroom and academic hub for cyber learning [12–17]. Second Life offers high degree of customizability and has opened rich avenues for entertainment, creativity, socialization, e-commerce, business, and education altogether on a single platform and thus promises to have the potential of shaping the future of the web.

Moving from 2D web environment to sophisticated 3D visual environment has raised concerns of accessibility. Unfortunately, a significant number of users are excluded from accessing the virtual worlds because of a disability [18–20]. Current virtual worlds do not provide features to allow individuals with severe visual impairments to access them using non-visual forms of feedback such as audio or tactile feedback. They lack textual representation [18] and so are not accessible to the visually impaired who often use screen readers for accessing other text based content. The US has an estimated 1.3 million individuals who are legally blind [21]. The social, entertainment and educational opportunities, and information sharing offered by virtual worlds could potentially benefit such individuals greatly as they often feel isolated and lonely [22]. Second Life offers social communities for users with disabilities. The Virtual Ability organization [23] offers communities such as Cape Able Island or the Heron Sanctuary [24] which are meeting places for the users with disabilities. Naughty Auties [25] is a
virtual resource center and meeting place for those with autism. But individuals with severe visual impairments are excluded from participating in such social communities. To make the use of virtual worlds comply with section 508 of the US Rehabilitation Act [26], it is important to investigate the possible solutions for making them accessible.

![TextSL Approach](image)

**Figure 1.2: TextSL Approach**

### 1.3 Proposed Solution

Visually impaired individuals most commonly use screen readers to access the web, so the approach discussed in this thesis dwells in the same vein. This thesis presents the development of a framework of a command based interface called TextSL that allows visually impaired players to access Second Life using a screen reader. This thesis focuses on the work taking Second Life as an example, since it is one of the most popular virtual worlds. But this work can easily be applied to access other open source virtual worlds using TextSL’s abstraction layer. TextSL primarily serves as a research platform which is independent of a specific virtual world implementation. TextSL
bypasses the Viewer and communicates with Second Life server to extract textual representations which can then be converted into synthetic speech using a screen reader (Figure 1.2). TextSL is written in C# allowing for it to be run on multiple platforms, as well as through a web interface using AJAX. TextSL supports a number of basic commands for navigation, interaction, exploration, and various advanced feature commands. Tutorial and help features are provided with TextSL for user training. Using this basic application, a number of user studies were also performed to identify further scope of improvement.

TextSL needs to be able to extract useful textual information from Second Life. One of the problems with large amount of user generated content in Second Life is that it lacks accurate and relevant meta data. Objects in Second Life can have a name and a description, and since most content creators think that the users can see what the object is, they leave these properties to their default value as “object”. Currently there is no provision in the Second Life viewer to enforce users to specify a useful name and description when they create objects. As TextSL relies upon textual representation of objects, this becomes a serious barrier in the approach of making Second Life accessible using synthetic speech [27–29]. Similar to how web images lacking alt tags remain invisible to the screen reader user, so are the objects named “object” are meaningless to them. The meaning of the web image can still be gathered from the text around the image or the filename [30], but the objects in virtual worlds do not have any such contextual information either. Adding this meta information manually to the millions of objects that lack useful name can be costly, time consuming, tedious, and error prone. Apart from this, there is no business model in place to fund such an activity.

The way the objects are defined in Second Life, can be exploited to do a large scale automatic object recognition using an appropriate shape descriptor [31, 32].
Chapter 1. Introduction

which can be constructed and compared efficiently by discriminating between similar and dissimilar objects. Categorizing Object using Bag-of-Features approach [33–36] along with vocabulary building using primitive decomposition can help in identifying the objects automatically on a larger scale. For example if an object is made up of 4 wheel like structures, and various seat structures, most likely it should be a car. A confidence level can be attached for this kind of an anticipation. For such an automatic labeling to work, a library of training examples for different object categories is required. A set of reliable and accurate meta data for objects needs to be created manually for this purpose, as the content in Second Life is inherently unreliable.

There are some similarities and dissimilarities between labeling objects in virtual worlds and labeling web images that lack alt tags. While the objects in virtual worlds can be treated with automatic labeling, it might be difficult for the web images as objects unlike the images are defined in isolation and represent one context, whereas the images can contain multiple subjects. Also, the objects have a topology, size, and orientation which can be exploited to form an algorithm that can automatically recognize them. The commonality is that both image labeling and virtual world labeling can benefit from utilizing human skills for recognition. Humans significantly outperformed the computers in recognizing images and involving humans in labeling images is a form of human based computation [37, 38]. The “games with a purpose” GWAP paradigm [39] aims towards making computationally intensive tasks easier by exploiting humans skills and making it attractive to them by presenting it in the form of a game. This approach has been successfully used for labeling images [40, 41], and transcription of ancient text[42]. This approach can as well be applied to the unique context of labeling virtual objects for creating the training examples which is needed for automatic labeling as mentioned above. The thesis presents a GWAP paradigm game “Seek and Tag” in
Second Life, developed to label objects in-world, which when fed to TextSL makes it meaningful to the screen reader users. Seek and Tag game is played using a scripted Object HUD developed using LSL, designed along the lines of Scavenger hunt game which motivates the users and makes it interesting for them to label objects via playing the game. This scripted object allows the labeling information to be stored in an external database which TextSL can query and thus better describe the Second Life world to the visually impaired players. The taxonomy of objects can further be built using this approach which can help in aggregating the the plethora of surrounding information to prevent overwhelming him/her.

1.4 Thesis Sections

The remainder of the thesis is laid out as follows: Chapter 2 provides a background of related work in this area. Chapter 3 provides a detailed description of the proposed TextSL framework. Chapter 4 provides a detailed study of meta data in Second Life and presents Seek and Tag Game to generate training examples for automatic labeling. Chapter 5 presents the results of the user studies conducted for both TextSL and Seek and Tag. Chapter 6 presents the observations made and problems encountered, pointing out areas of future work and the thesis is concluded in Chapter 7. Towards the end Chapter 8 provides with a glossary of most commonly used terms and definitions in virtual worlds.
Chapter 2

Background and Related Work

The interaction in virtual worlds is similar to that of other video games. It is typically modeled after third person games. A number of games have been developed for the visually impaired individuals and an extensive list can be found at http://www.audiogames.net/.

2.1 A Survey of Accessible Games for the Visually Impaired Individuals

This section presents a survey of various approaches that have been used to make games accessible to the individuals with visual impairments and whose interaction closely resembles the virtual worlds.

- “Shades of Doom” [43] is the first visually impaired-accessible, first-person-shooter (FPS) game modeled after the popular FPS game Doom (Figure 2.1). The main character is moved using cursor keys while audio cues such as footsteps, the
sound of wind are used to help find the way through various levels. A navigation tool called the “environmental analyzer computer” provides audio descriptions of the environment.

Figure 2.1: Shades Of Doom

- “Audio Quake” [44] is a first-person-shooter game based upon the Quake game. AudioQuake (Figure 2.2) replaces visual feedback with “earcons”, which are structured sounds that obey musical conventions. They are designed to alert the player
to an object or an event, although they do not sound like their referents in the real
world. Stereo sound is used to convey the spatial information. Sounds get louder
as a player moves closer to an object. Arrow keys are used to move and turn with
some customizable options to help players control their characters.

- **“Terraformers”** [45] is a first person shooter / adventure game which is accessible
to both sighted and visually impaired players (Figure 2.3). It uses 3D sound
response system. The main character can either be navigated using the visual
display (which also has a high contrast setting for the players having low vision)
or using the sound. The game uses audio cues to assist the user such as a tone
that represents “North” and different variations of tones are used to represent
eight compass points. The player can tell what is in front of the player by using
sound radar. By using a “ping” it is possible to tell how far are the objects from
the front of the character. Hitting a key on the keyboard, it is also possible to
tell what type of object is in front of the character, using a voice playback system.
Using the combination of the voice responses and the tone responses, it is possible
to successfully navigate a 3D world.

- **“Powerup”** [46] is a multiplayer educational game developed by IBM that sup-
ports various accessibility features for visual, motor, and cognitive impairments
(Figure 2.4). Visually impaired users can play this game using built in self-voicing.
Audio cues such as footsteps provide additional guidance. It has four different is-
lands with four different challenges to complete. Visually impaired users have the
same options as sighted players and can issue a number of commands such as “look
left” activated by key presses to get the information about their environment.
All these accessible games, except “PowerUp”, borrow their game play from their original versions and use the same way of interaction, but have been specifically developed to accommodate visually impaired users. The main strategy used in these games is transforming visual feedback into a form of audio feedback, either through self voicing, ambient sound, or sonar based technique such as earcons. To allow for techniques such as speech or audio cues, these games require some form of augmentation. For example in Shades of Doom, audio cues are augmented to the objects, and Terraformers and
PowerUp provide textual description for objects and locations in each level. Compared to their original versions, the levels in these games have been designed to have only few objects or avatars of interest around the player, to avoid overwhelming the player with a lot of information. There are other strategies of making blind accessible games such as converting visuals into haptic but are not the focus of this body of research.

### 2.2 Virtual World Accessibility

Virtual worlds have enjoyed increasing popularity over the past years but they currently are not accessible to the visually impaired people. The engagement and social activities offered by them can benefit these individuals a lot. The virtual worlds are vast and there are numerous object types with the large amount of user generated content. This limits the possibility of using the above mentioned strategies to the virtual worlds. As the virtual world content is owned and created by the users, modifying the content to provide audio or text is technically not feasible as one cannot modify the content if he/she does not own or has modify permission on them. Yet another feature of virtual worlds is that it lacks any kind of combat, which is the key feature of the other games that requires them to be able to respond fast. Thus, many of the mechanisms used such as sonar or haptic feedback that would allow identification of enemies may be useful but do not apply to the virtual worlds that well. Moreover augmenting the Second Life environment could change the experience for the sighted users which might be undesirable. There have been efforts towards addressing the issue of virtual world accessibility and following are few approaches that have been researched for making them accessible to the visually impaired:
• Second Life viewer was open sourced in 2007 and which enabled many developers from around the world to enhance its features. As the underlying infrastructure cannot be changed, a post hoc approach needs to be adopted to make it accessible. A modified version of Second Life viewer has been developed that allows for visually impaired users to navigate their avatar using force feedback where different objects can be distinguished through different vibration frequencies [47]. But this approach again is restricted only to distinguish a few categories as the vibration frequencies needs to be memorized.

• A guide dog project [29] developed by US based Virtual Ability group offers visually impaired users a guide dog object that can be worn by the user in the Second Life world. It provides services such as navigating the user to a particular location, querying environment information, etc. as the user issues commands using Second Life’s chat interface. The object provides feedback using synthetic speech. The guide dog object is a scripted object developed using Linden Scripting language. The functions offered by this approach is very limited and only focuses on providing a basic form of navigation and exploration. If used from within the viewer it allows for hearing in-game audio, but built-in speech synthesis typically does not provide customizations as the screen readers. Moreover, a sighted user might be required for the initial setup of the object.

• IBM’s Human Ability and Accessibility Center is also working on an accessible web based client for Second Life [28], which can be accessed using a screen reader. This client provides basic navigation, communication and exploration functions with the use of hotkeys (Figure 2.5). A hotkey based interface is again limited in supporting a large number of different actions on a large number of objects and avatars.
As social interaction is found to be players most favorite part in the MMOs by a study of virtual worlds [6], an accessible solution should cater this need. Though Second Life client offers voice chat, most users use text chat to interact with other avatars. Taking all this into account accommodating the existing viewer does not seem to be a good solution, so we developed TextSL that adopts a radically different approach using screen readers which may potentially serve the accessibility issue better. Visually impaired people use screen readers for various applications and usually they tailor screen readers to their preferences something that self-voicing games may not support. One of the limitations of screen readers is that it can only provide speech from one source, whereas virtual worlds typically produce feedback from different modalities (visual/audio) and different sources (avatars/objects) simultaneously. Dealing with
multiple types of feedback can be a topic of further research. TextSL also provides a solution that extracts meaningful information linearly and and at the same time accepts input iteratively. Also, TextSL is able to calculate a navigation path which prevents the user to be stuck at some obstruction in-world.

There are many similarities between TextSL and the above mentioned techniques, yet TextSL is significantly different and has much more to offer. The guide dog object also uses a command based interface and IBM’s client also uses a screen reader. But both clients provide limited functions when compared to TextSL as they do not allow object interaction, collision free navigation or a mechanism that prevents overwhelming the user with audio feedback.

Relevant meta information of the objects in the virtual world environment is the key towards making any of these text based approaches successful. As virtual worlds contain a huge amount of user generated content, the associated meta data usually does not contain meaningful information for the visually impaired users. Adding meta data is one of the prerequisites to make this approach work. Adding meta data to the virtual world objects is closely related to the approaches that aim at adding alternative text to the web images.

### 2.3 Image Labeling

Several important online applications such as search engines and accessibility programs for the visually impaired require accurate image descriptions. The lack of meta data in virtual worlds is similar to the problem of lack of web image description and can derive an insight from the solutions seeked for the image labeling. Two approaches of Image labeling can be identified:
Automatic Labeling: [30] Web images can be labeled based on heuristics or modeling methods such as semantic concepts and statistical relationships between images and words. Automatic labeling can be efficiently applied on a large scale, but the accuracy cannot be trusted. Mechanisms to add supplementary manual metadata are sometimes required for practical deployment [49].

Manual Labeling: This allows adding external metadata to the web images. This is usually done to create training data for automatic labeling.

Several manual and automatic methods have been developed over the years:

- The ALIPR (Automatic Linguistic Indexing of Pictures - Real Time) [50]: It is a machine-assisted image tagging and searching service developed at Penn State University for fully automatic and high-speed annotation of web images. The system uses a categorized image database for training the algorithms and annotates any online image specified by its URL. This work proves that algorithms can be developed to annotate general photographs with certain accuracy by learning from a large collection of example images.

- WebInsight for Images [51] The system is developed to improve the accessibility of web images by automatically formulating alternative text for them and inserting it into the pages that users view. WebInsight consists of three modules based on contextual analysis of linked web pages, enhanced Optical Character Recognition (OCR), and Human labeling. This mechanism combines both automatic and manual labeling. The system caches alternative text in a local database and can add new labels after the web image is downloaded by the user for the first time.
Automatic techniques used to categorize images are inadequate, largely because they assume that image content on a web page is related to adjacent text. But the text near an image is often scarce or misleading and can be hard to process. Computer vision cannot yet accurately determine their content and there are no guidelines about providing appropriate textual descriptions for the millions of images on the Web. Manual labeling can be relied upon and is the most accurate method for obtaining precise image descriptions, but it is a tedious and labor-intensive process and is extremely costly. Moreover, most people do not find manual labeling engaging enough to do it. Games can be used to make such tedious tasks enjoyable. Louis von Ahn spearheaded the “Games with a Purpose” (GWAP) [39] paradigm. Following games have been developed on these lines to make the web images accessible:

- **ESP Game** [41]: The ESP game accomplishes the task of image labeling through a simple online game that randomly pairs players together for 2.5 minutes who see common images one by one in this duration. Figure 2.6 shows a snapshot of this game. The games’ goal is to guess what label your partner would give to the image, and if both the players enter the same word for the image on the screen, they score points, and the next image is displayed. The goal is to agree with the partner on words for as many images as possible within the time limit.

- **Google Image Labeler** [52]: (Figure 2.7) It is developed to allow users to label images and help improve the quality of Google’s image search results. It is very similar to the ESP game except that the results of the partner’s guesses about the images are displayed at the end of the game to help player learn the game better. While ESP gives a total game minutes of 2.5 minutes, Google image labeler gives 2
minutes and along with that there is also a timeout for each image. Both ESP and Google Image Labeler provide a “pass” option, for the partner, for each image.

- Phetch [53]: It is a 3-5 person game (Figure 2.8) which is developed to generate explanatory descriptions for randomly chosen images from the web. Players are randomly grouped with others from the web. Initially, one of the players is chosen randomly as a describer, and the others are the seekers. The describer gets an image and gives a description of it. The seekers with only the text information
from the describer, must find the image using a search engine. The first seeker to find it wins and becomes the next describer. Each session of the game lasts 5 minutes, during which all players should go through as many images as possible. The web image data generated from this game is proven to be more accurate for the visually impaired web users.

2.4 Virtual World Object Labeling

IBM’s Human Ability and Accessibility Center developed a web based interface to Second Life to address the accessibility issue \[28\]. In addition to that, IBM provides a scripted object for the Second Life Viewer which allows users to add descriptive information to Second Life. A special annotation device is equipped on the avatars of the sighted users. When an object in Second Life is clicked, a web browser (Figure
Figure 2.9: IBM’s Annotation Web Page

2.9) is invoked allowing sighted users to add attributes such as a custom name, a short and long description of items or locations. The information is then saved in an external database and IBM’s client interface can retrieve these object descriptions when a user encounters objects with missing meta information. This information can improve the meta data in virtual worlds as it allows sighted users to provide detailed meta information. However it could be a tedious and time consuming task to provide such information for millions of objects. Everytime an object is to be labeled, a new web browser window is invoked which makes the already tedious task less user friendly, and more tedious. Moreover some firewalls can block such a transaction and this approach might involve security issues. The approach presented in this thesis allows for large scale labeling of virtual world objects. It uses in-built communication protocol of Second Life to avoid the possibility of the user’s machine’s security configuration problems.
Chapter 3

TextSL

With the increasing popularity and the huge success of the virtual worlds, it is quite evident that the next stage in the web’s evolution will incorporate elements from three-dimensional virtual worlds [2]. Virtual worlds like Second Life provide an immense level of interactivity and customizability and provide an altogether different experience for activities like socializing, entertainment, creativity, distant education, trading, playing games and more [15–17]. Since it is owned and created by its residents, it provides a huge number of possibilities and opportunities that can be done inside the Second Life world. Almost anything that a person can imagine, can be made possible in the Second Life World. It has taken the limited 2D interaction of traditional video games to 3D interaction and exploration domain, which provides much more rich and real-world experience.

Taking all these into consideration, Linden Lab, the company behind Second Life, made the Second Life Viewer open source to allow Second Life to grow and become reliable and scalable. This initiative was made to allow deeper collaboration between the industry and community, and advice the development of market-driven
standards [54]. The potential of Second Life is vast and the way in which residents wish to use it varies enormously. This provides immense opportunities for residents, many of whom are adept developers and have built profitable businesses around the Second Life Grid. Linden Labs have been very open-source friendly, and wanted to give the community a way to participate and improve the way they create software, with the hope that this will accelerate the innovation on the Second Life grid, enabling anyone to enhance the software and its architecture in all sorts of ways, making Second Life a much more compelling experience. Linden Lab has also given tacit approval to an open source project, libOpenMetaVerse (previously known as libSecondLife), to reverse engineer the Second Life protocols with the main aim of making an open source interface library for Second Life [55]. All these efforts and opportunities offered by Second Life and Linden Research have attracted academic and technological interest from developers, researchers and universities from all over the world into this project.

Looking at the vast scope of Second Life and the way the world is responding towards its future development [56, 57], the need for making it accessible to the individuals with disabilities requires equal attention. TextSL is such an effort towards making Second Life accessible to the individuals with visual impairments. TextSL is a command based interface which enables visually impaired users to access Second Life with the help of a screen reader. This work in this chapter has also been partly published as a paper in the Assets’09 Conference [58].
3.1 Design Requirements

3.1.1 Audio Feedback

The design requirement for making Second Life accessible to the users with visual impairments is to transform visual feedback from Second Life into a form of audio. Most games surveyed in Chapter 2 use the approach of giving some kind of audio cues to make the games accessible to the players who are visually impaired. Different types of sound feedback like sound radar, earcons, and ambient spatial sounds are made available to identify or locate enemies in these games. But the virtual worlds genre is fundamentally different and has a plethora of different objects and avatars. While ambient sounds or earcons make it easy to distinguish a limited number of objects or enemies which fit the requirement of the aforesaid games, this approach can not scale enough to identify a wide variety of objects, avatars and actions possible in virtual worlds. Moreover non-game-like virtual worlds are created and owned by their users, so it’s not possible to augment objects directly to the audio cues or text.

All the entities (objects, avatars, sims, islands, etc.) in Second Life can be given a name and a description, so synthetic speech can be produced to describe such an environment [27–29], which is essentially converting the name and description text to the audio feedback as per the requirement. Synthetic speech can be provided either by using self voicing API's within TextSL or by using a screen reader. However, screen readers are more flexible in terms of the functionality they offer. Users with visual impairments use screen readers for various applications such as word processors, emails, spreadsheets, web browsers, etc. They may prefer a particular screen reader over other and often tailor the settings to their personal preferences such as rate or pitch, something that the self voicing solution may not support.
3.1.2 Text-based Interface

One of the limitations of using screen readers is that they provide speech from one source only, whereas virtual worlds provide feedback from multiple modalities (visuals/audio) and many sources (objects/avatars) simultaneously. So the two main requirements from TextSL design are to be able to extract meaningful textual information from Second Life in linear form and at the same time be able to accept input iteratively.

Virtual worlds have their origins in Multi-user dungeon games (MUDs) [59]. MUDs allow players to explore dungeons, fight monsters, and interact with each other using a command based interface that iteratively accepts commands from the user and provides text output. MUDs have their roots in single player text adventure games like Zork [60]. Zork offers a number of interaction mechanisms that can be used in the proposed text interface client for Second Life. These interaction mechanisms in MUDs and text adventure games like Zork have proved to be fun to play and are efficient and engaging regardless of the command based interface. Furthermore, they cater to the constraints posed by the use of a screen reader. TextSL thus can utilize a text based interface model based on an interface similar to the Zork game. Figure 3.1 shows a screenshot of the Zork interface.

Modifying the Second Life Viewer to accommodate such a solution was considered. The Second Life grid is divided into regions and the Second Life Viewer requests the region server to send the environment identifiers of the region after logging in. But since Second Life Viewer’s functionality is mainly concentrated with rendering, this would prove to be an overhead and an inefficient solution. So for extracting textual information from Second Life, building a lightweight interface would suffice the
Figure 3.1: Zork

design requirement. The interface can utilize LibSecondLife to craft such a solution.
LibSecondLife is a software library that can be used in any third party application to
communicate with the servers that control the virtual world of Second Life. The library
maintains compatibility with the Second Life protocol and can be used for creating client
bots and automations in the virtual worlds which use the Second Life Protocol. Thus,
leveraging this library to connect and extract information from Second Life servers, can
provide most of the functionalities that TextSL requires.

3.1.3 Implementation

TextSL needs to provide a simple solution which the users with visual
impairments can operate without assistance. To accommodate this requirement, TextSL
does no graphics rendering; it is light weight and can be run on low-end machines or through a web interface. It does not need to be updated with every new version of Second Life Viewer and does not go through any tedious installation process either, which makes it easier to use for the players with visual impairments. Moreover, TextSL is platform independent application, developed in C#, and does not require any platform specific libraries. Developing a stand-alone application also has an advantage of serving as a virtual world research platform for the project of making Second Life accessible.

3.2 TextSL Functionality

TextSL is a command-based interface, which accepts certain commands and performs the corresponding actions accordingly. Furthermore, it continuously updates the user with real-time changes in the Second Life environment surroundings. TextSL can be run mainly in three different modes of operation: Normal, Advanced, and Chat. These modes of operation are explained later in this section.

3.2.1 Getting Started

All the users interested in accessing Second Life must sign up for a Second Life account. The sign up process involves several steps, including the customization of the visual appearance of avatars. These are the challenges which a user with visual impairment would be required to circumvent before being able to access Second Life. Secondly, after signing up, when the user signs in for the first time he or she is automatically teleported to the Orientation Island where the user is expected to finish Second Life orientation, which is designed for sighted users and is neither feasible nor useful for an individual with visual impairments to perform and complete.
Fortunately Virtual Ability Inc. [23], a non-profit corporation based in Colorado, USA, provides a simpler web form which an individual with visual impairments can easily use to get through the sign up process using a screen reader. The user must choose a unique first name and last name from a list of available names in Second Life for an avatar; provide a password, date of birth and an email address. To register the user must also pass an audio CAPTCHA test. After signing in for the first time through this process, the user’s start location is set to Virtual Ability Island, a virtual community in Second Life for users with disabilities. All the help and links pertaining to getting the user started with TextSL are available at http://textsl.org/.

3.2.2 TextSL : An Interpreter

TextSL is a command based 2d interface. The users can be easily trained to learn the available commands by using its tutorial and help features. TextSL acts as an interpreter that allows the user to issue commands using natural language. For example, if a user wants to walk towards a certain object or an avatar, or give something to an avatar, he or she can issue commands like: “go to <object-name>” or “go to <avatar-name>” or “give <object-name> to <avatar-name>”. These commands are intuitive and are based on natural language. Alternatively, he or she can also issue a command which is synonymous: “move to <>”, “walk towards <>”. All the different action based commands in TextSL have been mapped to commonly used synonyms just like natural communication. The use of synonyms makes the commands easier to use and remember, and thereby allows a user to process a variety of such actions on a large number of objects and/or avatars in Second Life more efficiently. TextSL parses the different parts of the command issued. It can distinguish between the action requested, the message, names of objects/avatars, the conjunctions, prepositions, and the adjectives. Examples of such
commands include: “sit on the chair”, “move north 10” or “move to the chair”. TextSL also allows the user to issue commands using the first letters of the commands to improve user efficiency as users get comfortable using the commands.

3.2.3 TextSL Commands

TextSL supports a variety of commands which can be categorized as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accession</strong></td>
<td></td>
</tr>
<tr>
<td>Login, SignIn</td>
<td>Logs in to the Second Life server. To login to Second Life, the user types “login &lt;first-name&gt; &lt;last name&gt; &lt;password&gt;”. After the first login TextSL remembers the user’s login information by saving it in an encrypted ini file. On the next login the user can just type “login” to login as the avatar he/she previously logged in with.</td>
</tr>
<tr>
<td>Logout, Quit, Exit</td>
<td>Log out and quit from Second Life. To logout and quit from Second Life, the user needs to type one of these commands: “quit”, “logout”, or “exit” if in default mode. Exit has dual purpose depending on context which is explained in “exit” command definition.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exits from the current mode to default mode. i.e. Exits from either the tutorial mode, test mode or chat mode to default mode. If running in default mode, logs out and quits. To exit the user types in “exit”.</td>
</tr>
</tbody>
</table>

**Exploration**

Continued on Next Page...
Table 3.1 – Continued

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe</td>
<td>Tells the location and lists the number of objects and avatars found within a specified range around the user’s avatar. All the objects and avatars within 360 degrees are considered. “Describe” is a iterative command and gathers more information on details as issued. The user types in “describe” to gather information about the location, objects and avatars near him/her. The user can also type: “describe objects”, “describe avatars” to know the names of objects and avatars nearby respectively. The user can further type in “describe &lt;object-name&gt;” to know further details on a specific object like if there is a sale on the object, what other functions the object provides, its description etc.</td>
</tr>
<tr>
<td>Move</td>
<td>Moves the avatar towards another avatar or object, or in some direction: west, east, north or south. The user types in “move to &lt;avatar-name&gt;”, “move to &lt;object-name&gt;” or “west/east/north/south distance in meters”. If the user does not specify distance, the avatar by default moves 5 game meters. If the user does not specify direction and just types move, the avatar by default moves 5 game meters towards north.</td>
</tr>
<tr>
<td>Fly</td>
<td>The avatar starts flying. Stop is a generic command which when applied to the current action stops it, in this case the avatar stops flying. To start flying the user types in “fly”. To stop flying the user types in “stop flying”.</td>
</tr>
<tr>
<td>Where</td>
<td>Finds out the current direction of an object or a building or an avatar relative the the user. To find out where an object, building or a person is, the user types “where is &lt;object/person name&gt;”.</td>
</tr>
<tr>
<td>Teleport</td>
<td>Teleport to a location. To teleport to a location, the user types “teleport to &lt;location-name&gt;”.</td>
</tr>
<tr>
<td>Follow</td>
<td></td>
</tr>
</tbody>
</table>

Continued on Next Page...
Table 3.1 – Continued

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop following</td>
<td>Starts following an avatar. Stop is a generic command which when applied to the current action stops it, in this case stops following an avatar. To start following an avatar, the user types “follow &lt;avatar-name&gt;”. To stop following an avatar, the user types “stop following &lt;avatar-name&gt;”.</td>
</tr>
<tr>
<td>Locate</td>
<td>Gives the coordinates of where the user’s avatar is. To locate the coordinates of his/her avatar, the user types in “locate”.</td>
</tr>
</tbody>
</table>

**Expression Actions**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crouch</td>
<td>Makes the avatar crouch. To crouch the user types in “crouch”.</td>
</tr>
<tr>
<td>Jump</td>
<td>Makes the avatar jump. To Jump the user types in “jump”.</td>
</tr>
<tr>
<td>Stand</td>
<td>If the avatar is sitting, makes it stand. To stand, the user types in “stand”.</td>
</tr>
</tbody>
</table>

**Communication**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Say</td>
<td>When an avatar writes a chat message, Second Life displays “&lt;avatar-name&gt; says &lt;message&gt;” in the chat box. So this command is introduced in line with that. The user Says a message for everyone in the neighborhood to hear. To say a message to everyone near his/her avatar, the user types in “say &lt;message&gt;”. Or simply begins the sentence with a double quote.</td>
</tr>
<tr>
<td>Whisper</td>
<td>Whispers a message to an avatar. To whisper a message only to a specific avatar, the user types in “whisper to &lt;avatar-name&gt; &lt;message&gt; ”. This starts the user’s session with the avatar he/she wants to communicate with and in the proceeding messages, the user can just type “whisper &lt;message&gt;” to whisper to that avatar he/she started the session with.</td>
</tr>
<tr>
<td>Chat</td>
<td>Enters into chat mode. In chat mode anything that a user types is not parsed and is understood to be just a chat message. To enter chat mode, the user types “chat”. To exit from chat mode the user types “exit”.</td>
</tr>
</tbody>
</table>

Mute

Continued on Next Page...
### Table 3.1 – Continued

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop mute</td>
<td>Mutes an avatar. Stop is a generic command which when applied to the current action stops it, in this case unmutes an avatar. To Mute an avatar, the user types in “Mute &lt;avatar-name&gt;”; to Mute all, types “mute all”. To unmute the user types in “stop mute all” or “stop mute &lt;avatar-name&gt;” whichever applicable.</td>
</tr>
<tr>
<td><strong>Interaction</strong></td>
<td></td>
</tr>
<tr>
<td>Sit</td>
<td>The avatar sits on the ground or on an object. To sit on ground, the user types “sit”. To sit on an object, the user types “sit on &lt;object-name&gt;”</td>
</tr>
<tr>
<td>Touch</td>
<td>The avatar touches an object. To touch an object, the user types “touch &lt;object-name&gt;”.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td></td>
</tr>
<tr>
<td>Help</td>
<td>The user types “help” to get a list of all available commands and modes or types “help &lt;command-name&gt;” to get detailed help of a specific command or mode.</td>
</tr>
<tr>
<td>Tutorial1</td>
<td>Teaches the user the basic commands of navigation and communication in Second Life in a controlled environment designed for the tutorial.</td>
</tr>
<tr>
<td>Test1</td>
<td>Tests the user’s basic command knowledge of navigation and communication in a controlled environment designed for the test.</td>
</tr>
<tr>
<td><strong>Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>Sets the Range of how far the avatar wants to see, how many objects and people he/she wants to see at a time. The user types in “range &lt;distance&gt; &lt;number of objects&gt; &lt;number of people&gt;”. This is available only to the advanced user. The default range is: 30, 10, 5 respectively.</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets the current user login information. Used when the user wants to login as a different user. To reset current login information the user types in “reset”.</td>
</tr>
</tbody>
</table>
All the TextSL commands are case-insensitive. TextSL also stores the history of typed commands and offers to set the Page-Up/Down keys to navigate the list of previously typed commands just as is typical of command-line prompts. Once a command is entered, TextSL parses the command, and with the help of the libSecondLife library protocol, sends the request to the Second Life server. The Second Life server sends back the feedback on the action performed which is received by TextSL, which provides the information back to the user.

The user can take tutorials and tests that TextSL offers to get acquainted with its functionality anytime and as many times as desired by the user. Tutorials and tests are stored in XML format. The XML files contain all the instructions and the command validation logic for any tutorial or test. Any tutorial or test can be designed if an appropriate XML for it is created. TextSL reads this XML and consequently gives one or more instructions to the user for the task to be performed. The instructions for the tutorial are tailored to be informational whereas for the tests the instructions are interrogative. The commands that the user enters in response are validated in either case. The test is controlled by a time factor as opposed to the tutorial which provides the user with unlimited time and number of attempts. However both are based on the same underlying basic design. TextSL also logs in all the typed commands and received response with a timestamp, in a logfile for later evaluation.

Once the user logs in, TextSL provides auto feedback that he/she successfully logged in along with a welcome message. TextSL also does an auto describe of the environment location, numbers of objects and avatars nearby. The user can then issue commands to navigate or talk to some avatar or perform any other actions. After navigating, an auto describe is always performed by TextSL to let the user know of his/her changed neighborhood/location in the SL environment. TextSL also provides feedback
Chapter 3. *TextSL*

if the user failed to move to a particular location due to some obstruction or permission issues.

### 3.2.4 Findings

While implementing the basic functions of TextSL some important observations were made:

- Objects in Second Life can be given a name and a short description, but often they are left to their default names (e.g., Object, component, shape, etc), as the content creators think that the sighted users visually identify the objects. When TextSL retrieves the name and description of objects to provide it to the TextSL users, these default and generic object names show up which provide meaningless information to the TextSL users. Lack of the meta information is a serious challenge to the goal of making Second Life accessible using a screen reader on the basis of textual representation of the SL environment. A solution to this problem is presented in detail in Chapter 4.

- Second Life’s rich environment is densely populated with objects. The density of the objects varies from region to region, but the regions which are popular and are points of interest are usually densely populated. On a sample of 433 regions, on an average 13 objects were found within a 10 meters radius of an avatar. This directly affected the implementation of a usable “describe” command. This presents the challenge of providing TextSL users with similar information as sighted users can see (up to 40 game meters), without overwhelming the user with a lot of information. An additional difficulty with such an object density is that the users can get obstructed by objects and get stuck while navigating and would
have no clue of what to do. To provide solution to these problems the following upgrades were done:

− Summarizer: The objects retrieved from the Second Life environment need to be processed before providing the user with the object’s information. The objects need to be prioritized in order to provide useful information to the user. First the objects are filtered based on a distance range from the user’s location (by default it is 30 game meters). The objects are then prioritized in the following order: 1) the objects are ranked according to their distance from the user, 2) the length of their name. It is assumed that longer the name the better is the description, but if it is too long (> 70 chars) then possibly it is just a sentence and not a good qualifying name and so it is ranked accordingly. The objects are sorted according to their rank and on querying the SL environment, the highest ranked objects(by default 10 objects) are provided to the TextSL user. The user can then iteratively use the “describe” command to get more information on them.

− Collision-free Navigation: The up-down-left-right arrow keys are mapped to navigate north-south-east-west. Second Life provides basic obstacle avoidance, but in case of larger obstacles at different angles, the user could potentially get stuck during navigation. To overcome this problem when a “move” command is issued, TextSL plans a route which is collision free using the A* algorithm [61]. TextSL issues commands to navigate the avatar to follow the new route. This solution was tested with certain locations and still needs further development to accommodate a generic The avatar is automatically teleported to the desired destination if the avatar does not arrive within a specified time limit.
Figure 3.2 shows a scene in Second Life and Figure 3.3 show the same scene as it is represented in TextSL. A beta version of TextSL was made available in November 2008 on the website http://www.textsl.org and currently has been downloaded more than 420 times. TexSL’s source code has been released under GNU Public License and can be downloaded from http://code.google.com/p/textsl/.

3.3 TextSL Architecture

TextSL leverages the libSecondLife library to communicate with Second Life. TextSL is developed using C# on a windows machine, but it is cross-platform though the current version only works with the JAWS screen reader [62]. TextSL implements a typical Service Oriented architecture. 1) When TextSL is issued a command to perform an action, it makes a call to the corresponding libSecondLife service which
Chapter 3. *TextSL*

Figure 3.3: TextSL

TextSL communicates to the SL server and provides feedback. 2) For getting notifications from the Second Life servers, TextSL subscribes to the desired services.

Following are the building blocks of TextSL:

- TextSL Interface
- TextSL Core
- TextSL Abstraction Layer
- LibSecondLife Layer
- A Screen Reader
The first three layers are tightly coupled to each other. These layers together with libSecondLife and a screen reader make a robust, multi-threaded textual interface to the Second Life world. LibSecondLife is the core library on which TextSL relies to be able to communicate and access Second Life servers. Below we present the detailed design (see Figure 3.4) of the libSecondLife library on top of which TextSL is built.

![TextSL Design](image)

**Figure 3.4: TextSL Design**

### 3.3.1 Libsecondlife

LibSecondLife is an open source ongoing effort to do a full reimplemention of the Second Life networking protocol. It is a .NET based Client/Server library which upholds compatibility with the Second Life protocol and can be used for creating clients and automatons in Second Life, OpenSim, or other virtual worlds which use the Second Life protocol. It provides a networking (protocol) abstraction layer to Second Life using
C# which offers better Windows compatibility. The library is designed for everything from low level packet construction to high level abstractions of avatars, simulators, and more. It is able to fully decode the packets from SL that describe assets such as prims, textures, inventories, etc. It also enables the creation of all such assets, as the Second Life Viewer does. In theory it gives developers access to the full power of the Second Life official client. The project itself is geared towards creating a standard API that any programmer could make use of for their own application.

Libsecondlife runs in a .Net virtual machine. Therefore, the only dependency it requires to run is .NET (in Windows) or Mono (in Linux, Unix or Mac OS X). There are many potential uses of libSL such as building independent clients, import/export back up Tools, client side plugins, testing of the Second Life platform, NPC style bots, etc. It makes a good choice for the TextSL client to serve as a textual virtual world interface.

Due to trademark/licensing issues, the libSecondLife library [63] was renamed as libOpenMetaVerse [55], and all the newer version releases conform to it, but the core library remains the same. TextSL still references the libSecondLife library but the new releases will be upgraded to libOpenMetaVerse.

LibOpenMetaverse consists of several .NET Assemblies (the following text is derived from their website [55]):

- OpenMetaverse.dll The core library which contains all the client functionality for accessing the Second Life, OpenSim and Simian servers. This is originally the core libSecondLife library.

- OpenMetaverseTypes.dll A 3d specific Types and Math library
• OpenMetaverse.StructuredData.dll A serialization library which implements full support for LLSD and Json for structuring data with loose typing useful for serialization of data over the capabilities system

• OpenJpeg A .NET wrapper for encoding and decoding Jpeg2000 data utilizing the openjpeg library

• OpenMetaverse.Http.dll - A CAPS Client and Server used by the OpenMetaverse library and Simian

TextSL uses only the core libSecondLife.dll to access Second Life and it currently does not need the rest of the assemblies, some of which are rendering and 3D specific. Here are the main classes of libSecondLife which TextSL utilizes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecondLife</td>
<td>libSecondLife’s main class which contains all of the functions that are required for the client/bot to communicate with the Second Life grid. LibOpenMetaVerse has renamed this class to GridClient. This class contains several classes designated to perform various tasks which are listed below.</td>
</tr>
<tr>
<td>NetworkManager</td>
<td>This class contains all the methods and callbacks to connect and disconnect to a sim and instantiate the next events.</td>
</tr>
<tr>
<td>AgentManager</td>
<td>This class which contains all the data related to the agent (the user avatar) and methods to perform actions on the agent, initializes the callbacks, and publishes events for the agent avatar(Self).</td>
</tr>
<tr>
<td>ObjectManager</td>
<td>All the objects in Second Life have a Unique Identifier (UUID) within the world. This class contains all the methods and events related to the objects in the Second Life grid. An instance to this class manages creating, rendering and all other functions of the objects.</td>
</tr>
<tr>
<td>AvatarManager</td>
<td>This class represents information management for other avatars.</td>
</tr>
</tbody>
</table>

Continued on Next Page...
### Table 3.2 – Continued

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParcelManager</td>
<td>This class contains information of parcels of land and the actions that can be performed on them.</td>
</tr>
<tr>
<td>AppearanceManager</td>
<td>This class contains all the functionality pertaining to the appearance of the avatar.</td>
</tr>
</tbody>
</table>

#### 3.3.2 TextSL Interface

The TextSL Interface takes the textual input commands from the TextSL user, and gives it to the TextSL Core and gets back the textual updates and feedback messages to be displayed to the TextSL users.

#### 3.3.3 TextSL Core

TextSL Core consists of a parser, a processor and the data in the memory. The processor manages all the data in memory. The Parser parses the input commands to extract the action required and sends them to the TextSL Abstraction layer to be executed. The processor gets the response and updates from the Abstraction layer, processes and sends them to the TextSL Interface. The processor is responsible for storing and managing the response data in memory. The processor goes back and forth with Abstraction layer, when required, before sending the response to the interface. For example when a user issues a “move” command, the Core first queries the Abstraction layer and determines if the user might get stuck. The Core then recalculates a collision free route and issues the new modified commands to the Abstraction layer accordingly. Everytime TextSL is launched, the processor loads the tutorial/test related XML data
and settings related data in the memory. The processor mediates between the interface and the Abstraction layer and manages all in-memory data. Figure 3.5 depicts the TextSL architecture.

### 3.3.4 TextSL Abstraction Layer

The TextSL Abstraction layer talks to the LibSecondLife layer to get access to the Second Life world. It performs two basic tasks:

- Makes appropriate calls to the libSecondLife layer when the TextSL Core requests to perform a task based on the command issued: these are called Actions. Some of the Actions are Login, Describe, Teleport, Move, etc. These actions make appropriate calls to the corresponding methods of LibSecondLife which are designated to perform the given task.

- Get the updates and feedback from the libSecondLife layer with all the real-time information of the Second Life Grid: these are called Listeners. Some of the Listeners are: Objects, Avatars, Chat, Teleport, Whisper etc. Here is a brief summary of what they do:
  
  - Objects: Listens to the updates and gets the object information nearby the avatar’s current location. The following lines of code explain this:

```java
client.Objects.OnNewPrim +=
new ObjectManager.NewPrimCallback(Objects_OnNewPrim);

client.Objects.OnObjectProperties +=
new ObjectManager.ObjectPropertiesCallback(Objects_OnObjectProperties);
```
“client” is an instance of “SecondLife” (or GridClient) Class. Whenever the user logs in or moves to a new location, the client fires “OnNewPrim” and “ObjectPropertiesCallBack” events to which the object’s Listener has subscribed. And thus is able to get all the notifications. Similarly all the other events which the TextSL needs, can be subscribed in the form of a listener.

– Avatars: Listens to the updates and gets the avatar information nearby to the avatar’s current location.

– Teleport: Listens to get the feedback on avatar’s teleport success or failure. Teleport also has an action which is invoked when when the “teleport” command is issued. Certain tasks can have both Action and Listener as desired.

– Chat: Listens to the chat channel and gets updates on any messages that the avatar/objects in the neighborhood might be saying.

– Whisper: Listens to the personal chat channel, if an avatar personally communicates to the user.

![TextSL Architecture](image)

**Figure 3.5:** TextSL Architecture
3.4 Summary

All the commands which TextSL facilitates are mapped to the Actions which are essentially the calls to the libSecondLife APIs. On the other hand, Listeners are the delegates which subscribe to the events of libSecondLife. LibSecondLife in turn accesses the Second Life servers to be able to get notified with the updates and feedback which are further processed by the TextSL Core and sent to the interface.

Whenever a session is established between a client and the Second Life grid, the Second Life server sends the information packets of just the nearby parcels (around 40 m radius: that a sighted user can see in the SL viewer). TextSL also receives only the surrounding information at a given time from the region server. This does not overload the client and makes the process efficient.
Chapter 4

Seek and Tag

While developing TextSL, it was discovered that a lot of objects surrounding the player lacked meaningful meta information. So TextSL filtered such objects because they do not make any relevance to the visually impaired players. But this is not a solution to the actual problem. Lack of meta data could prove to be a serious threat to the text based approach of making Second Life accessible via a screen reader. To verify the magnitude of this problem, object data from various islands on Second Life was collected and statistically analyzed. This was done with a tweaked version of TextSL, named as TSL DataServer NPC. NPCs are nothing but non-playing characters in virtual world/role playing games, which are controlled by an automated program and not by humans. They are often developed as bots designed for a specific purpose.
Chapter 4. *Seek and Tag*

4.1 TSL DataServer NPC

4.1.1 Functionality

On running the TSL DataServer NPC, the bot auto logs in as a non-playing character, continuing from its last saved location in Second Life. It takes the bot about 5 minutes to retrieve the object data around him. After every 5 minutes, he is programmed to save the collected object data into a local SQL database and teleport himself to a randomly selected new location. This time spent by the bot at one location can be altered through the configuration settings. The Second Life grid is divided into small chunks of lands called regions or sims. There are currently 13,543 regions in Second Life world [64]. The names of these regions are stored in an XML file, which the DataServer NPC uses to randomly teleport and collect object data. TSL Dataserver NPC subscribes to the notification updates to retrieve data from the Second Life grid using the libSecondLife
library, in a similar fashion as TextSL. The following object information can be retrieved and saved in a MS SQL database using TSL Dataserver NPC:

- **UUID**: Id which uniquely identifies the object in-world.

- **LocalID**: Object’s local Id in the sim where it is located.

- **Name**: Object’s name.

- **Description**: Object’s description, if any.

- **Parent**: An object is created by linking a group of basic objects called prims. Each object group has a parent prim.

- **X, Y, Z Coordinates**

- **Type**: One of the seven basic prim types such as cube, torus, sphere, etc.

- **Size of the Object**: Size of the prims that constitute an object. This information is collected for future analysis.

- **Region information**: ParcelID and name, SimID and name, total area of the parcel in game square meters, total prims on the Parcel, maximum number of prims allotted to the Parcel, total number of prims in the Sim, maximum number of prims allotted to the Sim. This identifies the location information where the object resided when this information was collected. (A Sim/Region is nothing but a collection of Parcels of land.)

### 4.1.2 Architecture

TSL DataServer NPC architecture is based on the TextSL architecture itself. The difference between the two is that TextSL Dataserver NPC does not have an
interface and the screen reader hook up; the Core only supports “login” and “teleport” actions and has an extra data layer to communicate with the local database to store the object information. The building blocks are as follows:

- TextSL Core
- Object Data Layer
- Local Database
- TextSL Abstraction Layer
- LibSecondLife Layer

Figure 4.1 gives an overview of the design. When the application is executed, it reads the configuration file to retrieve the user login information and sends login request to the libSecondLife layer. Once the login is successful, a session is established. The core subscribes to the object update notifications using the abstraction layer. The Core then reads an XML file which has the extensive list of regions or sims in Second Life. It randomly picks a region, and teleports there and stays for about 5 minutes. The Core listens to the object data updates, retrieves the required object information and starts storing it in memory. Few test runs revealed that the optimum time required for the NPC to retrieve the object data around him is about 5 minutes. The user can change this time duration through a configuration setting. After 5 minutes, the Core asks the Object data layer to save the collected data in memory into the local Database. Once this is done, the Core requests to teleport to the next randomly selected sim and repeats the process. This way the DataServer NPC keeps itself logged in and collects and stores object data from different islands.
4.1.3 Data Collection Experiment and Results

Five TSL Dataserver NPCs auto logged in and teleported themselves into a sample of 433 randomly selected regions and collected this object data over a week’s period. Complex SQL queries were run on this database to analyze the data collected. Through this experiment it was found that 31.3% ± 2.02% (SD=21.4%) of the objects in Second Life are named “object”. These objects were mostly buildings, furniture, pets, billboards, trees, bushes which are in abundance in Second Life. This number varied significantly from region to region and was as high as 85% in one of them. Further analysis revealed that non-descriptive names like “shapeblock”, “Box 40x40x0.8” are amongst the most frequently used names. These names may seem relevant to the content creators but are meaningless to the TextSL screen reader users. Furthermore, there were some objects which lacked a relevant name because of improper grouping or linking of prims. For example, a house is made up of different prims that are linked together to form a single house object. But instead there were examples wherein these prims were not linked together into a single house object but left into 2-3 parts having individual names. Based on this, an estimated 40% of objects in Second Life lack meaningful meta information. This motivated us to step back and seek a solution to this problem before proceeding any further.

4.2 Labeling Objects

It is evident that adding meta data to the objects in Second Life is a prerequisite to make it accessible to the screen reader users. This can be done in two ways:
• Manual Labeling: Second Life has millions of objects. Although manual Labeling has the benefit of adding descriptions to the objects, it can be a tediously huge and time consuming task and also error prone. Moreover, in Second Life the objects in-world cannot be renamed by the avatars who do not own them or if the objects do not have modify permissions. So this is not feasible technically unless all the owners go back to edit all of their owned content and rename appropriately. This cannot be enforced and so is not a possible solution.

Figure 4.2: Automatic Labeling using Object Categorization

• Automatic Labeling: Automatic Labeling can be done by using an appropriate shape descriptor which can be constructed and compared efficiently while discriminating between similar and dissimilar objects [31, 32, 65]. Firstly, Second Life objects have a topology and size and are composed from smaller seven different types of solid object entities called prims which are defined analytically with permutations such as twist and taper. Secondly, Second Life objects are modeled after real world objects, so the apparent homogeneity of the objects can be exploited. These properties of Second Life objects can be utilized to perform automatic labeling on a large scale. Figure 4.2 depicts this classification. To make this approach work, we took into consideration the following:

– Object Categorization is the key towards handling a large number of objects with a common structure. This technique in contrast to object recognition,
which identifies the object to be recognized from a database of known objects, is not only efficient but provides the advantage of recognizing previously unseen objects. But one of the main challenges to do 3D object categorization is to find an appropriate shape descriptor which can distinguish between similar and dissimilar objects. Typically 3D objects are represented by “polygon soups” - unorganized and degenerate sets of polygons lacking any topology or solid modeling information [32]. Prim based composition of the objects in Second Life makes it easy to map them to physical models than meshes. As a proof of concept, a suitable shape descriptor was chosen and it was able to discriminate between a small number of object categories efficiently.

Figure 4.3: The Big Picture

- Training Data: To perform object categorization on a large scale, a library of training examples is required. But second Life meta data is highly unreliable and cannot be used as such. To obtain a large set of reliable training data,
a GWAP paradigm [39] based in-world game called “Seek and Tag” was designed. The relevant meta information thus created as a result can be stored in an external database in real-time and TextSL can retrieve this information and provide a meaningful feedback, about the Second Life environment, to the screen reader users. Since one of the design requirements was to keep TextSL light weight and security/configuration hassles free, it was not suitable to let TextSL communicate to this external database directly. Rather the communication is facilitated by the use of a non-playing character bot designed for this purpose. TextSL can communicate with one of the pools of TSL Agent NPCs (explained in the Design section), using the existing instant messaging protocol of Second Life, which can update and fetch object data from the database. Figure 4.3 depicts the entire flow diagram.

4.3 Seek and Tag Game

Creating a training database of objects manually is again tedious and error prone. Designing a GWAP-based game can facilitate labeling in-world and motivate players to play and contribute at the same time. Only sighted users can play this game and in the process help solve the accessibility issue in Second Life.

4.3.1 GamePlay

Seek and Tag is a GWAP-based game developed in lines of the popular “Scavenger Hunt” game. In Scavenger hunt, players seek to gather a list of specific items and the goal is usually to find the list of items first to win the game. Whereas in Seek and Tag, the player is given the task of seeking an object that falls in a proposed
category and tag it under that category. The purpose of the game is to be able to divide objects in Second Life into a list of defined categories, and involve fun in this labeling/categorization task so as to motivate the players to do it. It is single player game, and the challenge is to tag as many objects as possible in the given time to score better. The reward system incorporated makes the game challenging and fun to play.

The game begins with a countdown of a total of 300 seconds. Each task of seeking and tagging an object under a category is to be completed in maximum 30 seconds to earn points. For each seek and tag the player 50 points. If the player finishes the task before 30 seconds, along with the 50 points, he is also rewarded with the leftover time which is cumulatively added to the total time left for the game. The faster the player tags the objects, the more time the player earns to seek and tag the remaining objects. As the player runs out of time, the final score is recorded. Figure 4.4 shows a snapshot of an avatar playing Seek and Tag in-world.
4.3.2 Design

The game has two main elements. 1) The Seek and Tag HUD, an in-world scripted HUD object, that provides the player with the interface to seek and tag the objects under a given category and incorporates gameplay logic through a script. 2) TSL Agent NPC, a bot which keeps itself logged in and is designed to listen to the object categorization information from the Seek and Tag HUD and store it in an external database. Once an object is tagged, the HUD uses instant messaging protocol to send the object categorization information to the TSL Agent NPC which stores it at the backend.

Seek and Tag HUD is a group of five basic prims linked together, with one flat board structure and four spheres. The player wears the Seek and Tag object as a HUD to be able to play the game in-world. When the player wears this HUD, the board and the spheres are located at the center of HUD screen. The board displays the task to be performed, score and time ticks. Each of the four spheres, called the “Object Representative Spheres” (ORS), are programmed to hover over the nearest four objects as the player moves. As the game begins, the player is asked to seek an object within a given time. A task is displayed on board, asking the player to find an object that fits a category. The avatar moves around in-world to search for an object that can be tagged under the asked category. As the avatar moves around, the four ORS hover over the nearest four objects and display their existing name. An ORS has all the information of the object on which it hovers, and acts as a representative of that object. Once the player finds a suitable object, he/she clicks the corresponding ORS to tag it. On being clicked, ORS asks for confirmation, if the player confirms, ORS sends a message to the TSL Agent NPC with the object categorization information. This message is
wrapped around a unique encrypted envelope so that TSL Agent NPC can identify that the message is sent by the HUD and can safely store the information in the database.

The design details of Seek and Tag HUD and the ORS are discussed in detail herewith.

### 4.3.2.1 Seek and Tag HUD

Seek and Tag HUD is an interactive object having scripts written in Linden Scripting Language (LSL). To better understand the design of Seek and Tag scripts, let’s take a brief look at the composition of a typical LSL program and its architecture. LSL is interpreted and executed on the Second Life servers (sims) and not the client (Viewer). Although the script editor is a part of the SL Viewer, the script itself runs on the server, which sends the results over the network to the Viewer. LSL emphasizes on “States”, “Events” and “Functions”. Many real life behaviors can be modeled with “states” and the same is true for the LSL programs. For example, a door can be either closed or open, representing its 2 states. An event can be thought of as a “Trigger”. Events are not user defined in Second Life, but rather predefined. When an event is triggered, it calls a routine called an event handler. The events are either triggered by objects and avatars interacting in the world, or they are invoked from within a script. For example, when an avatar touches an object, a “touch_start” message is sent to the object, which causes the “touch_start()” event handler of the object to execute. A minimum LSL program has one state with one event handler in it. Functions lay inside of the events. There is repository of built-in functions which the programmer can use out of the box. Functions can also be user-defined and can be written by the programmer to perform desired tasks. Scripts can make an object perform a lot of tasks and Seek and Tag HUD is such a scripted object.
The scripts in Seek and Tag HUD follow a typical Master-Slave architecture. The Master Script resides in the “board” which commands the slave scripts residing in the “ORS” to perform tasks like hovering over the objects. The slave scripts in return give the feedback to the master script when an object is tagged and the master script in turn processes the data to update the score. Figure 4.5 depicts this interaction. Following are the building blocks of the Seek and Tag HUD:

- Task and Score board: This is a prim of a board structure which is rendered on top of the player’s head as HUD. The board displays the current task, time left for the current task, game time left and scores earned by the player and gets updated as the game progresses. The Master script embedded with this prim takes care of all this functionality behind the scenes. Here is the pseudo logic for the Master Script:

  – Start-Stop Game: An LSL event handler gets triggered on the board’s click event to start or stop the game. When the board is first clicked, it triggers the game to start, whereas on the next click the game is forcefully ended.
– A Task Pool: from which the script picks the object categories.

– A Count-down Timer: to calculate and display the time left every second during the game.

– Sensor: is an event exposed by LSL which when subscribed to, retrieves the object and avatar information near the player in real-time. This can be configured to retrieve environment information within a given range, angle and time interval. As the sensor senses the objects around the player, the event handler gets triggered and is programmed to retrieve the object coordinates and other properties of the nearby objects.

– Command to Hover: As and when the board sensor retrieves the information of a nearby object, it commands one of the Object Representative spheres (ORS), via an LSL prim linked message, to hover over the object. It sends the object coordinates and other object properties such as UUID, name description etc in this message. At a time only four objects can be hovered upon, as there are only four ORS prims in the HUD. These 4 ORS act as representatives of the objects on which they hover upon.

– Command to Restore: Commands the Object Representative sphere, via a LSL prim linked message, to move back to the center of the HUD screen.

– A Score Calculator: listens to the success message from Object Representative Spheres to update the score. The board prim receives a message from the ORS when an object is tagged, updates the scores, and displays the next task from the pool of tasks.

• Object Representative Spheres (ORS): Seek and Tag consists of four sphere prims which are rendered on top of the player’s head (at the center of the HUD screen) before the game begins. As the game starts these spheres are commanded by
the Master script to hover over the four nearest objects to avatar. As the name suggests, each of these spheres contain the object information and serve as the representation of those objects in-world over which they hover. ORS are required to implement the tagging feature because there is no other way in Second Life LSL for an avatar to find out that which object was clicked by him/her. Each ORS contains the slave script to be able to do the tasks as commanded by the Master Script. Here is the pseudo logic for the Slave Script:

- Hover Over Object: On receiving the “Move” command from the Master script, the ORS is required to move over the object coordinates specified in the message. This object position is relative to the camera position. So ORS script converts it into the HUD Coordinates and calls the appropriate LSL function to move itself to the new position.

- Restore to Center: On receiving the “Restore” command from the Master script, the ORS script calls the appropriate LSL function to move the ORS back to the HUD center.

- Confirm Object Tagging: On-click event handler of ORS implements the functionality of opening a dialog to get confirmation from the player if it wants to tag the current object under the current task category. If the player confirms, it sends a message, using instant messaging function of LSL, to the TSL Agent NPC. The message contains the object information and the category under which it was tagged. This message is wrapped around by an encrypted identifier so that TSL Agent NPC can identify the message. It also sends a feedback message to the master script that the object was tagged successfully.
4.3.2.2 TSL Agent NPC

- **Functionality:** On running the TSL Agent NPC, it auto logs as a non-playing character into its last saved location in Second Life. This bot is designed to listen to the instant messages sent to him by the Seek and Tag HUD. It identifies the message identifier to verify the source of the message and then saves the object categorization information sent via this message to a local database. TSL Agent NPC can also be queried by sending an appropriate message containing object UUID. On receiving such a query, the Core sends a request to the data layer to fetch the object information for the requested UUID(object) and sends back a message containing the object information. This message is also wrapped by an encrypted identifier to avoid spamming. This way, TextSL can be programmed to query details of a particular object through an instant chat message to the TextSL Agent NPC. TextSL agent NPC can then query the database and send the details of the object using the instant chat message of Second Life.

- **Architecture:** TSL Agent NPC architecture is based on the TextSL architecture itself. The difference again is that it does not have an interface and the screen reader hook up, whereas the Core only supports login and instant messaging actions and has an extra data layer to communicate with the local Database to store the object categorization information. Figure 4.6 refers to this design. The building blocks are as follows:

  - TextSL Core
  - Object Data layer
  - Local Database
  - TextSL Abstraction Layer
Figure 4.6: TSL Agent NPC Design

- LibSecondLife Layer

When the application is executed, it reads the configuration file to retrieve the user login information and sends login request to the libSecondLife layer. Once the login is successful, a session is established. The core subscribes for instant messaging notifications using the abstraction layer. Once a message is received, it verifies that the message is wrapped by an encrypted identifier to make sure that the source of the message is the Seek and Tag HUD Object. It then saves the object information including the UUID, localID, name, description, coordinates, Size, object type, taper, twist, region information in which the object resides, the object category under which the object was tagged, and the avatar information who tagged the object along with a timestamp. To refrain the players from randomly
clicking objects and cheating, tagging attempts by one player is saved and verified with tagging attempts by other players for the same object.

The overall function and the flow diagram can be seen in Figure 4.7.

This game is the first step towards establishing a training data by creating a set of object with accurate names/categories. These set of objects can be classified into object classes which will help in building a taxonomy for virtual world objects. This will not only facilitate automatic labeling but will also providing information in a more useful and digestible form to the TextSL users.
Chapter 5

User Studies

Two user studies were conducted to validate the approach, and evaluate the performance and degree of success of TextSL and Seek and Tag in achieving their respective goals. This was done in collaboration with our research group and also appears in the thesis “Towards Generalized Accessibility of Video Games for the Visually Impaired” [66]. Following are the results from these user studies:

5.1 TextSL User Study

Second Life Viewer provides various functions to access Second Life world including exploration, communication, interaction, and content creation. TextSL supports these basic functions except content creation. The focus of this user study is to evaluate whether the specific mechanisms used in TextSL allow screen readers users to access Second Life. Following are the specific goals considered:

- Accessibility: The study seeks to derive the answers to the questions of ‘Is TextSL accessible using screen reader?’ ‘Can the user control the avatar viz. communicate,
explore and interact in world using TextSL?", ‘Is the amount of feedback provided by TextSL overwhelming to the user?’.

- Usability: The “Beyond Accessibility to Efficiency” (BATE) principle states that the goal of assistive technology should be more ambitious than simply to provide access [67]. So this study should also seeks to evaluate usability in addition to accessibility. This includes the evaluation of learnability, efficiency, memorability, errors and satisfaction. These are some of the qualitative attributes proposed by Nielsen [68]. To do such a qualitative analysis, TextSL needs to be compared with Second Life Viewer for which sighted users need to be involved. The questions that the user study considers include: ‘Are sighted users able to perform same tasks on both clients?’, ‘Are sighted users able to achieve same goals within similar time frame and with similar efficiency?’, ‘Is the TextSL solution acceptable?’ and further suggestions to improve the solution.

5.1.1 Design

Four visually impaired participants and eight sighted participants between the age of 21 and 51 years were recruited for this user study. All the participants had prior experience with computers and with playing games. All the visually impaired users used screen readers on a regular basis. The study was conducted on an isolated private island similar to the other islands in-world. This was done to avoid any interruptions from other avatars, have a controlled environment with accurate object names, and to design a setup that can help setting up the study according to its specific needs. There were two setups, one designed for the users to get familiar with accessing Second Life features and the other to test if they were able to learn and access Second Life. The user study at both setups was designed to have identical set of tasks to be performed.
and had a similar spatial layout to be able to perform optimal comparison. Table 5.1 gives an overview of this user study.

**Table 5.1: TextSL User Study**

<table>
<thead>
<tr>
<th>User Study Steps</th>
<th>SL Viewer User</th>
<th>TextSL User</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tutorial:</strong></td>
<td>The participants first receive a tutorial on how to move their avatar, explore the world, interact with other avatars and objects after logging in Second Life. The avatar of the participant is being observed by another avatar who facilitates for responding to the communication during the tutorial. This was performed at a random location by the sighted users. The participant was also asked to log all the tasks he/she performed in a paper so that a comparison on the usability can be done later.</td>
<td>The visually impaired users adjust the screen reader (JAWS) settings to their personal preferences before starting the tutorial. TextSL was programmed to provide a similar tutorial which would facilitate the users to get familiar with the commands which they can use to explore and query the environment, move their avatar, interact with other avatars and objects. They can try to learn how to perform each task with unlimited tries and time. Once a task is performed successfully, TextSL interpreter can recognize it and it then moves on to teach the next task. Another avatar was observing the actions of the avatar of the participant logged in through TextSL and was there to respond to participant’s communication during the tutorial. All the commands the participant typed in the in the TextSL interface were logged in a log file by TextSL along with a timestamp. This was performed at setup-1 by visually impaired and sighted users both.</td>
</tr>
</tbody>
</table>
Table 5.1 – Continued

<table>
<thead>
<tr>
<th>User Study Steps</th>
<th>SL Viewer User</th>
<th>TextSL User</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test:</strong> The test was designed to ask the users to perform the tasks they learnt in the tutorial. The test asked the avatar logged in from TextSL to first move to a location and identify a number of objects near him/her. Then move to another location and interact with objects nearby, and then move to yet another location and chat with an avatar there.</td>
<td>—</td>
<td>Once the participant finishes the tutorial, his/her avatar is automatically teleported to setup-2 which is designed for the test. He/she is asked to perform a series of tasks, each having a pass condition to gauge that it was successfully performed and a limited time to finish. During the process the user is able to issue any commands, including “help”, but to go to the next task he/she needs to enter the correct pass condition designed for that task within the time limit or else the task times out and the next task is asked. The tasks were designed independently in a way that failing one task would not affect the other. All the commands the user typed in the TextSL interface were logged in a log file by TextSL along with a timestamp.</td>
</tr>
<tr>
<td>Play with the Environment</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Fill the questionnaire at the end which would help in qualitative analysis of the solution.</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

5.1.2 Results

All the participants successfully completed all the tasks they were asked to perform. The evaluation was based on the following:
• Quantitative analysis: Following data was collected: Total time it takes to perform the tutorial and all the tasks, number of tasks successfully completed, total number of commands used, and the number of unrecognized commands. Specific screen reader settings were also recorded.

• Qualitative analysis: This was collected through a subjective feedback from the questionnaire. Visually impaired users were asked to provide their experience, suggestions, and the problems in general. Sighted users were asked to do a qualitative comparison of both the clients. It is difficult to directly compare the two clients, as screen reader users cannot use the SL Viewer and sighted users may not be proficient screen reader users, so only the applicable factors are taken into consideration during the analysis.

Following are some of the results derived from this user study:

![Figure 5.1: Average Task Performance Times](image)
• Efficiency: In virtual worlds time may not be a critical factor, but still it is a good measure for efficiency. Figure 5.1 shows average performance time for each task using both the clients by the sighted and visually impaired users. The average execution time to complete the tutorial for TextSL was lower. The sighted users were observed to be familiar with Second Life Viewer type of an interface and were found to be playing around within the world, which explains the greater time. Interaction and navigation with TextSL were found to be significantly slower. This is mainly because TextSL is iterative and Second Life Viewer is not. This can be because the Second Life Viewer can navigate and at the same time see what all objects are around them, whereas TextSL users first type the command for navigation, then type the command to describe the environment, so on and so forth iteratively. Communication was similar for both TextSL and Second Life Viewer as it is iterative in nature. Social interaction has been identified to be the most common activity in virtual worlds [6], in which TextSL performs equally good. Thus, even if TextSL’s less efficient interaction is not optimal, but is still acceptable.

• Memorability and Errors: On an average sighted users used the “help” command once during the test. Visually impaired users did not use the “help” command during the test, which could imply that they were able to memorize the commands during the tutorial better than the sighted users. The number of errors users made were not significantly different between the two clients. The errors made with the TextSL included unrecognized commands, typos, or commands without required parameters, whereas the errors in Second Life Viewer included clicking on wrong object because of the vicinity. The errors are inherently different in nature and so do not qualify for comparison as such.
Usability: This was gauged in terms of the overall experience including the factors of learnability and satisfaction through a questionnaire in which users were asked to rate on a 5-point Likert scale. Figure 5.2 shows a relative comparison of the usability between the clients. Some users found the information overwhelming while others found some feedback too little, for example one of the users said, “help command should be shorter”, while the other said, “provide names of objects without typing the command again”. Feedbacks for improvement were taken into consideration for example “Only one task can be done at a time”, “can not move diagonally”, “adding shortcuts” etc. Further studies could determine a more digestible and optimal amount of feedback.

The development of TextSL and the results acquired provided valuable insights into making virtual worlds accessible and led to the identification of issues in the text based approach, mainly the lack of meta data.
5.2 Seek and Tag User Study

Seek and Tag game is designed to be able to create training examples that can be used for automatic labeling of objects in Second Life. This user study was conducted to evaluate Seek and Tag prototype’s effectiveness, accuracy and usability as compared to manually providing names to the objects. The collected object labeling data is associated with a confidence level that indicates the level of confidence of an object’s labeling information. If an object is manually named with the same name by (n) users then (n) is the confidence level with which one can guarantee that the object is what its name suggests. On the other hand, an object needs to be tagged by (n-1) users to achieve the same confidence level. In general to achieve higher level of confidence more naming or tagging attempts must be exercised.

5.2.1 Design

The study was again performed at a private isolated island which was designed after a typical Second Life island using existing popular Second Life objects. There were two setups at location A and B consisting of 25 objects each. The objects were dispersed randomly in the area of 50x50 game meters. All the objects were named “object”. Eight participants between the age of 24 and 43 years were recruited for this user study. They were given a brief tutorial of the Second Life environment, how to do manual labeling, and how to play Seek and Tag game.

Eight users were asked to first manually label or name the objects either in location A or B. They could name the objects in any order they prefer. Then six of these users were asked to switch their locations and play Seek and Tag. This design was implemented to do a more fair comparison. Seek and Tag asked the players to find
objects iteratively. The users could tag any objects they find fit, but the tasks to find the objects were in the same order, to verify if the efficiency increases over time. Users could also name the same objects multiple times if they considered that suitable. Towards the end, users were asked to fill a questionnaire to gauge the usability of the game.

5.2.2 Results

Following are the results computed from this user study:

- **Accuracy**: Seek and Tag outperforms manual naming with high confidence intervals for more number of objects. Manual labeling may not come to a consensus as different users may name it differently whereas tagging assumes the name. Figure 5.3 depicts the results of confidence levels achieved with manual naming vs tagging.

- **Efficiency**: This is measured by the average time taken to either name or tag an object, to achieve a certain confidence level. It was found that for higher levels
of confidence, Seek and Tag was more efficient than naming. There is a tradeoff between efficiency and accuracy.

- The Average time to tag each object: This was found to be slightly decreased over the time indicating that as players get accustomed with the game environment they are able to locate objects faster and are motivated to achieve a high score.

- Usability: Usability was evaluated using a questionnaire which was designed to evaluate the interface, mechanics, gameplay, and the fun aspect of the game on a 5 point Likert scale. The game scored an overall average of 3.7 for the usability.

Seek and Tag proves to be an effective game. This is one of the first GWAP games that uses third person interaction mechanism and creates a more compelling experience. Several variations of this game can be easily developed with little tweaks in the gameplay to provide users with a variety of challenges and motivation.
Chapter 6

Discussion and Future Work

The development of TextSL identified some unique characteristics of virtual worlds and provided valuable insights and experiences for making virtual worlds accessible to the visually impaired individuals. Some topics for further discussion and several areas of future work were identified during the course. Here is a brief summary of these findings:

- Command-based interface Vs shortcut based interface: Which of the implementations is better is a topic that is open to discussion. The user study shows that the command based interface is inherently slower, however it allows for efficient interaction with a large number of different objects and avatars and various actions, which is difficult to support using shortcuts. Furthermore, synonyms of the actions can be mapped to the same action which not only improves the memorability but also facilitates interaction using natural language.

- Data Mining: There is a large amount of data found around the user in the rich Second Life environment, which when fed to the screen reader, can easily overwhelm
the visually impaired user. So the need for aggregating surrounding information and recognizing the objects of user interest becomes evident. TextSL offers a simple form of summarization but a more concise and richer form of feedback is needed. One of the research directions can be to explore abstraction mechanisms for data aggregation, at different levels of representation and make it more informative and descriptive. Furthermore, a narrative can be created using natural language.

- Multi modal interpretation: A lot of audio feedback can easily overwhelm the visually impaired players. A larger amount of feedback can be made digestible through a multimodal approach, for example, a screen reader can be used in conjunction with a tactile or braille display. TextSL can be developed to stream and support Voice over IP chat, and various music and sound feedback from the Second Life world. This would require some research on including cue sounds along with speech and finding how effectively can text data be aggregated to produce local sounds. Trade offs will be involved and experimenting this can give some interesting results.

- Large Scale Automatic Labeling: The Seek and Tag interface and TSL Agent NPC infrastructure for creating a training database is already in place. A reasonably sized set of training data is required to be created, which includes object categories to analyze the effectiveness of automatic labeling using shape descriptor approach.

- Object Taxonomy: Seek and Tag can be modified to be able to establish a taxonomy for virtual world objects. This could serve two purposes:
  - Help create object categories that are required for automated object labeling approach. Object categories need to have multiple examples of real Second Life objects that can be used to exploit the commonalities and develop a
suitable shape descriptor that allows for automatically categorizing unknown objects.

– Second Life world is densely populated, and an object taxonomy can help in providing a digestible feedback to the screen reader users, e.g. [bicycle, car bus] \rightarrow [3 vehicles]

The creation of such a taxonomy can be tied with the Seek and Tag game by making the game more challenging the longer the user plays it. Initially generalized and more common objects can be sought which are more easy to find and as the game progresses more specific categories can be sought. The game can also be tailored to less experienced players by providing extra gaming cycles which would train them and at the same time facilitate for more data labeling captures. Seek and Tag can start with highest level of categories, could create training examples and then split them into subcategories when possible. Shape descriptor can facilitate the criteria to split the categories. Some form of manual labeling will be required when a match cannot be found between the examples that establish this sub category.

• The user studies conducted on the island were restricted to an area of 50x50 game meters. Regions in Second Life can be as big as 256x256 game meters and seeking objects can be more challenging in such large areas as objects may be hidden and not in the field of view. Visual cues can be provided when the allocated time starts to run out for a given task, enabling the possibility of identifying the location of the object. Integration of new rules will be required to provide such functionality.

• Evaluation: Further work needs to be done in establishing a centralized process and an associated database that facilitate remote evaluation, local evaluation and user data collection.
• Converting text: A lot of text embedded in object textures (images) e.g. information board, or bulletin board, is not readable to the visually impaired users. Experimenting with some Optical Character Recognition (OCR) methods to convert this text into a stream of alphabets which can be fed to screen readers could prove to be a useful feature.

• Supporting various functionalities of Second Life: There are various functions which TextSL still does not support, including:

  – Research on how to create objects, which is mainly a visual activity, using TextSL.
  – Extended interaction with different interactive objects
  – Search engines of Second Life which facilitate searching sims, users, groups, events, etc.
  – Contact management: Friends request, group request and joining, teleport requests
  – Supporting Second Life sound feedback
  – Customizing avatars
  – Managing inventory

• Currently TextSL only supports the JAWS screen reader. Supporting various other screen readers such as Microsoft Narrator, Thunder, Windows Eyes, or Mac Voiceover is still required.

• Virtual classrooms: Second Life is frequently used as Virtual classrooms. A community for visually impaired people can benefit from such a platform. TextSL needs to be evaluated for being used by the visually impaired for learning through
virtual classroom setups in Second Life. They can also provide important inputs for further development needs through such a community.
Chapter 7

Conclusion

This thesis presents the results of development and evaluation of a command based interface for virtual worlds called TextSL. TextSL allows the visually impaired users to access Second Life using a screen reader. Using TextSL the individuals with visual impairments can navigate and explore the world and interact with other avatars and objects. User studies with the visually impaired as well as sighted users indicate that TextSL is easy to use and the functionality is comparable to that of the Second Life Viewer. The educational opportunities and the social benefits offered by Second Life can also be exposed to the visually impaired using this solution. Promising feedback was received on these aspects in the course of the research. Large amount of user generated content in Second Life has led to lack of relevant meta data. During the research this was identified to be the biggest hurdle in making the text-based approach a success. Automatic Labeling can be used for a large scale virtual world object labeling using the shape descriptor approach. This requires a set of training examples which can be provided by the Seek and Tag game, a GWAP-based solution of naming objects in virtual
worlds. Seek and Tag has proved to be effective and its mechanics can be implemented in any virtual world that lacks meta data.

The other issues identified were the overwhelming of the user with the densely populated surrounding information of Second Life world, and the inability to create content using TextSL. Future studies will focus on adding functionalities for the remaining features that Second Life offers, including challenges of content creation, interacting with scripted objects, allowing for a mechanism that will enable a descriptive and digestible narrative, large scale automatic labeling and creation of an object taxonomy. Learning from the user studies, it can confidently be inferred that with further research, virtual worlds like Second Life can be made more universally accessible. This thesis integrates elements of various fields such as computer graphics, computer vision, computer science, and games, applied to the domain of human-computer interaction. It also utilizes techniques such as “Games with a Purpose” and promises to provide, with further progress, a compelling impact on the user experience of the visually impaired.
Chapter 8

Glossary

Here is a glossary of selected terms and their definitions from the virtual world dictionary which will be useful for better understanding of the thesis. These have been withdrawn from Second Life wiki pages and other Second Life resources [3, 69–71].

- **Av, avatar, avie**: A character in Second Life, usually referred to as a person or a resident. A new resident is a newbie.

- **Chat**: Words typed in chat are visible to all people within 20 meters of the speaker. Optionally, the speaker may shout and the words will heard all the way across the sim in which the speaker is located.

- **Grid**: The basic structure underneath all of the Second Life world. The grid is subdivided into thousands of regions simulating physical surface geography. Linden Lab provides the grid and residents build the superstructure and surface infrastructure. Sometimes the grid goes down temporarily for maintenance.

- **Help Island**: A tutorial area in Second Life. Another is Orientation Island. Help Island (HI) allows activities to be selected and is not mandatory.
• HUD: Heads-up display is a visual display of preset options as part of the user interface (UI)

• In-world: Being inside or online in a virtual world

• Linden: An employee of Linden Lab, the company that produces Second Life. The entire LL staff takes the name Linden as a last name for their avatars.

• Linden Lab, LL: Common name for Linden Research Inc., the company that owns, controls and hosts Second Life.

• Lindens, Linden dollars, L$: SL money that has a fluctuating exchange rate like any other currency. The virtual currency can be traded for US$ or US$ can be used to buy L$.

• LindeX: The Linden Dollar Exchange, a currency exchange trading dollars for lindens and lindens for dollars.

• LSL: Linden Scripting Language used to program objects in SL.

• Mainland: Mainland is the land run by Linden Lab and associated tier fees are paid directly to them.

• Orientation Island: The tutorial area where all new Second Life users start out. Unlike Help Island, Orientation Island (OI) is directed and mandatory.

• Parcel: A piece of land can be split of joined into parcels. The smallest parcel you can create in Second Life is 16m (4m x 4m).

• Prims, primitives: The raw 3D objects from which everything in SL is constructed. A basic building and creation material. Linden lab provides these so anyone can build. Every object, clothes, houses, furniture, cars, plants, etc. are created from
prims by the people who use SL. All things are created from these basic building blocks. They come in several shapes. Each land area can support only a certain number of prims. That quantity usually is quoted for all items. For example, a basic 512 square meter plot of land will hold only 117 prims. This means a small lot will allow for only a small house with limited furnishings.

- Private islands: The islands that are privately owned and operated, separate from the mainland.

- Real life, RL: The natural world where we live when not visiting Second Life.

- Region: A simulation, as in a virtual region of the Second Life world that simulates a physical environment. A region, also called a sim, is an area hosted by a single server CPU. The SL grid is subdivided into thousands of regions. Each Second Life simulator is independent with its own characteristics, rules, ratings (PG or Mature), and themes.

- Resident: An avatar or character in Second Life. A new resident is a newbie.

- Rez, rezz: To create or spawn an object in the Second Life world. Also used to refer to the sharpening of the image seen on the computer monitor screen. Its a term from the movie Tron.

- Second Life, SL: the 3-D virtual world published and maintained on the Internet by Linden Research Inc. (a.k.a. Linden Lab) of San Francisco. It is built entirely by its residents.

- Sim: simulation, as in a virtual region of the Second Life world that simulates a physical environment. A sim, also called a region, is an area hosted by a single server CPU. The SL grid is subdivided into thousands of regions. Each Second
Life simulator is independent with its own characteristics, rules, ratings (PG or Mature), and themes. A virtual place hosted on a single computer CPU server.

- **Whisper**: Speech or chat text (usually by objects containing scripts) that is audible only within a 10 meter radius from the object. Compare with regular chat (20m radius), shouting (100m radius) and Region Say (sim-wide).
Bibliography


