A Platform for Reflective Decision-Making in a Virtual Environment

Robin Howlett

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science

2007

DECLARATION

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Robin Howlett

September 14, 2007

PERMISSION TO LEND AND/OR COPY

ACKNOWLEDGEMENTS

Many thanks are due to my supervisor, Dave Lewis, for his invaluable advice and for the benefit of his experience throughout the course of the project.

To my NDS classmates, thank you for making my experience in Trinity College one to remember.

Lastly, I would like to thank my family for their continued support over the course of my school and college years.

Robin Howlett

University of Dublin, Trinity College

September 2007

ABSTRACT

Organisations are adopting agile organisational structures, with flat hierarchies, a team-working approach and devolved decision-making, as well as becoming more customer-centred and knowledge-intensive. On-line communities are also exhibiting these characteristics and are progressing at an even faster rate. These highly-dynamic organisational entities need to be able to understand how to identify and resolve the conflicts resulting from decentralised decision-making. Providing this information however is a challenge as studying such decision-making is difficult because observing such distributed organisations in practice is expensive, time consuming and obtrusive for organisational members willing to test out such consensus reaching tools.

Therefore a collaborative Virtual Reality environment such as Second Life provides a good laboratory for observing how a distributed community would attempt to organise itself dynamically, collaborate to reach decisions and manage its own resources. By doing this, an organisation could discard the rigid, difficult and expensive requirements engineering phase and manage itself through distributing authority within its own community, resulting in a more cost-effective and flexible solution.

This dissertation describes a platform for reflective decision-making in a virtual environment; a unique combination of a community based policy management system component, a cognitive decision-making component and a conceptual 3D object used to interface to these functionalities through a wide variety of interactive techniques. Using a sophisticated virtual environment, the focus of the research is to investigate the effectiveness of combining the above functionalities within a highly-dynamic three-dimensional environment.

Table of Contents

LIST OF FIGURES

Chapter 1: Introduction

This dissertation describes a platform for reflective decision-making in a virtual environment. It has been designed as an investigation into the effectiveness of using a virtual three-dimensional environment to allow people/communities to collaborate to reach decisions and transfer that knowledge throughout the community, so that a clear view exists of how those decisions have been made.

This chapter is intended as a precursor to the main dissertation. In order to fully understand the dissertation research, design and implementation, an introduction to the main ideas and technologies involved is provided.

1.1 Why model reflective decision making?

Understanding how to identify and resolve the conflicts resulting from decentralized decision making is becoming increasingly important as knowledge-intensive, customer-centred organisations adopt agile organisational structures (flat hierarchies, team-working, devolved decision making) and as on-line communities (which naturally exhibit these characteristics) become larger and more important in people lives.

However, studying such decision-making is difficult because observing such decentralised organisations in practice is expensive and time consuming and people may indeed not wish to be observed to be willing to test out new consensus-reaching tools. Therefore, using a collaborative VR environment such as Second Life may provide a good laboratory for such studies.

1.2 Virtual environments in this context

Multi-Use Virtual Environments (MUVEs) are online virtual worlds and the term is most widely used to describe Massively Multiplayer Online Games (MMOGs) that are not necessarily game-specific.

A MUVE is an artificial test-space where the behaviour of people can be observed and potentially influenced. MUVE is “a highly constructionist environment as it forces the users to his/her own active approach in solving particular problems. Together with the natural support of socialization it absolutely corresponds with the current streams of educational theory” (Brdicka, 1999).

The rapid growth of social networks over the past decade has demonstrated that the general public has taken to this virtual community-based idea. One of the most attractive features of Second Life, certainly one of the most powerful, is the degree to which the virtual universe can be tailored using the underlying Linden Scripting Language (LSL). It has certainly added scores of amateur programmers to the ranks, many of whom had never touched a line of code before the first time they tried to make their own virtual house in SL.

The popularity of SL is also being used to form large online communities within the game. However the tools that are available to these groups fall short of the level of sophistication needed to managed a collection of “virtual people”, never mind about how they

1.3 Vision and Research Goals

Before pursuing any course of research it is necessary to clearly define research goals and discuss the motivation behind such goals. The main objective of the dissertation is to develop a tool in a virtual environment that will permit an online community to make decision about itself and come to a clear community understanding about how those decisions were made.

This will in fact incorporate a number of challenging concepts and technologies. The resulting platform should be able to recognise a variety of conceptual communication techniques that exist in the virtual world, interpret them accurately, process the requested functionality including resolving conflicts that it encounters and finally be able to do this without external interference i.e. it should be self-configurable and to be able to be managed by the community itself.

The design and implementation behind this application will be the most challenging as some of the most complex conceptual concepts like mental models, knowledge representation, knowledge transfer and cognitive decision making will all have to be abstractly modelled. They will then have to be able to dynamically interact with virtual objects within the environment in order to use these advanced communicative techniques to make decisions about which self-management functionalities to use. In fact, in many ways the nature of this dissertation is a proof-of-concept.

The system will have to be evaluated from the perspective of whether the users of the system can understand and use the system, in terms of functionality, visualisation and richness of perspective.

1.4 Document Structure

Chapter 2: State of the Art

Chapter 2 outline the current state of research in the fields of existing decision-making platforms and 3D/Virtual Reality tools and environments. The second section introduces the concepts of communities whose policy-managed system is self-managed and the notion of organisational decision-making is introduced. The concept, and importance, of knowledge chapter is introduced and current interesting research that involves the Second Life virtual environment is presented also.

Chapter 3: Design

This chapter discusses the design of the three components that together form the system application.

Chapter 4: Implementation

Chapter 4 explains how the components of the system are implemented, with any background implementation detail supplied. The chapter first presents the technologies used and gives full implementation details for each component.

Chapter 5: Evaluation

Chapter 5 presents the evaluation approach for the project and discussed the results from the evaluation. This chapter gives an overview of how the application was evaluated by four external participants.

Chapter 6: Conclusions

This chapter presents ideas for future development and research of the system followed by the final conclusions about the success of the project.

Chapter 2: The State of the Art

The purpose of this chapter is to outline the current state of research in the fields of existing decision-making platforms and 3D/Virtual Reality tools and environments. The research for this dissertation was interesting in that the subject matter contained within is in a brand-new area of research and therefore has been broken up into its component sections.

Each section presents the research conducted for this dissertation. All information in this chapter was gathered from white papers, documentation, websites, applications and public presentations.

2.1 Second Life (SL)

Second Life is currently the most popular MUVE in use today and since its launch in 2003 by Linden Research, Inc (Linden Lab) has maintained a consistently high profile in the mainstream media for the last two years.

The users of Second Life are termed *Residents* (or *agents*), and their appearance is their avatar, a virtual representation of themselves. The basic avatar is initially human in appearance, but every aspect of the avatar is customisable and can produce a wide variety of forms and manifestations.

2.2 Self-Managed Communities and Decision-Making

Traditional policy engineering approaches rely on a role-based model requiring precisely defined structure of roles and rules to be specified within a detailed requirements engineering phase (R Sandhu, 2000). This approach is static and subject to expensive and laborious requirements modelling.

2.2.1 Community Based Policy Management (CBPM) and Self-Management

CBPM is a scheme which “supports greater levels of flexibility, dynamism, decentralisation and autonomy in management” (Kevin Feeney, 2006). It is a policy-based management scheme that aims to remove external authorities in determining grouping abstractions.

CBPM uses the notion of *community* as the grouping abstraction with the aim of allowing groups within the organization itself to define communities to reflect the natural structure of the organization. Communities organise themselves into a hierarchy based on authority, with the *root* representing the entire organisation, meaning sub-communities are subject to the root’s authority. Communities can be granted authority to the organisation’s access resources, and as communities themselves are resources, authority can also be granted to spawn further sub-communities.

Policies in a community are responsible for ruling on which actions on which resources can be undertaken by that community’s members. By default, the root community consists of all members of its sub-communities. The organisation structure is modelled through controlled delegation of authority through the community hierarchy (Kevin Feeney, 2006).

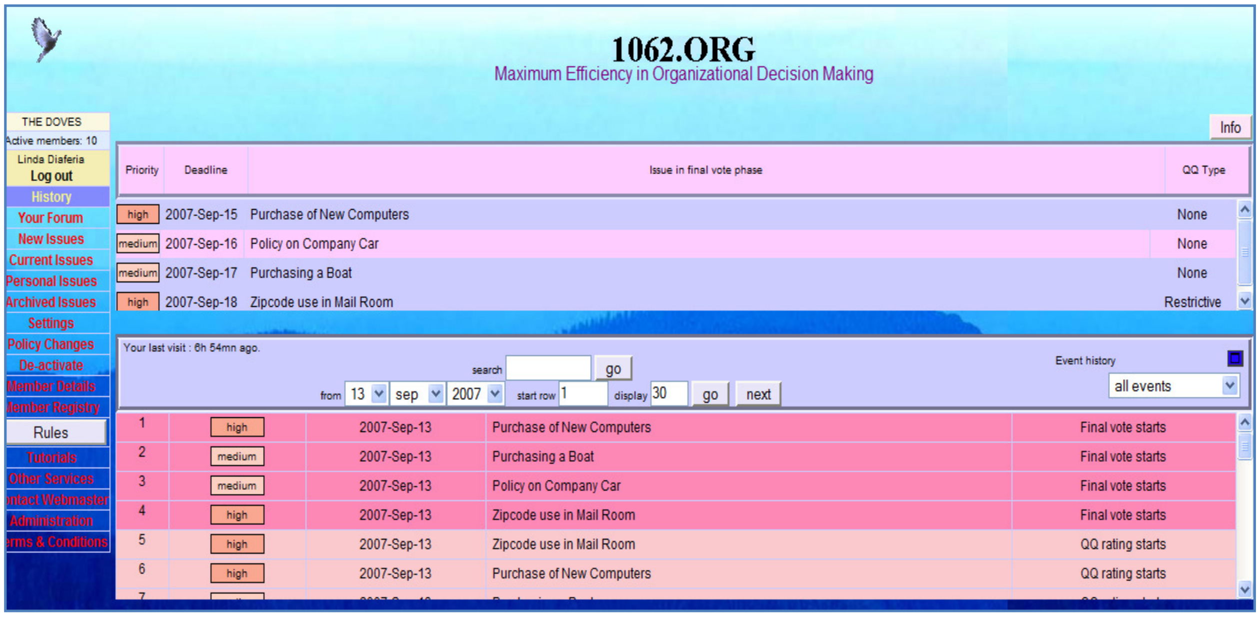
Self-management in a community means permitting members of a community to manage the community-based model themselves. Self-management is then enabled by distributing authority to manage resources in a controlled way and by enabling an organisation to manage its own security and access control policies.

The implementation of a self-managing organisation by a CBPM is described later in the *Implementation* chapter of this dissertation.

2.2.2 Organisational Decision Making

1062.org is a web-based service that aims to provide an "efficient issue and consensus generator" and markets itself as a "fully transparent decision-making voting system" (Quincy).

The website is a form-based implementation of direct democracy principles and as such is intent on being used to generate organisational policy. 1062.org introduces the idea of each proposition in each issue having its own discussion area within the site. Users that have subscribed to the service have full and equal access to all functions defined including the management of the very rules (through the manipulation of 62 settings) that govern the entire policy management process.



Priority	Deadline	Issue in final vote phase	QQ Type
high	2007-Sep-15	Purchase of New Computers	None
medium	2007-Sep-16	Policy on Company Car	None
medium	2007-Sep-17	Purchasing a Boat	None
high	2007-Sep-18	Zipcode use in Mail Room	Restrictive

Your last visit : 6h 54mn ago.

1	high	2007-Sep-13	Purchase of New Computers	Final vote starts
2	medium	2007-Sep-13	Purchasing a Boat	Final vote starts
3	medium	2007-Sep-13	Policy on Company Car	Final vote starts
4	high	2007-Sep-13	Zipcode use in Mail Room	Final vote starts
5	high	2007-Sep-13	Zipcode use in Mail Room	QQ rating starts
6	high	2007-Sep-13	Purchase of New Computers	QQ rating starts

Figure 1: 1062.org presentation front-end

1062.org however advocates the separation of the social needs of an organisation's environment from its business needs, and indeed changing the environment also. The service also details the various degrees of participation that can be undertaken by members of an organisation utilising the website's services.

1062.org basically acts as a virtual chairperson who aims to work around mutually arrived at dictates.

However, the weaknesses of 1062.org are the similar weaknesses of any two-dimensional tool that aims to work with organisations – it recognises that off-site discussion and decision-making needs to take place and implementations like this website simply cannot provide such functionality.

2.3 Knowledge Transfer

Knowledge Transfer is "the process through which one unit (e.g., group, department, or division) is affected by the experience of another" and "the transfer of organizational knowledge (i.e., routine or best practices) can be observed through changes in the knowledge or performance of recipient units. The transfer of organizational knowledge, such as best practices, can be quite difficult to achieve" (Argote & Ingram, 1999, p151) .

This subject of knowledge transfer in a virtual environment has yet to be researched so studies that dealt with it in real life are examined.

2.3.1 Analogical Training versus Individual Case Training

Analogical training uses comparison and general principles to transfer knowledge rather than individual case training which aims to acquire a similar previous experience and apply the same axioms.

However, there is a key problem with knowledge transfer; people tend to access previous knowledge that bears *surface* (not *structural*) similarity to the problem at hand, but when structurally similar problem solutions were presented, they were regarded as more useful (L. Thompson, 2000).

This discovery was made in an experiment designed to investigate whether knowledge gained from case studies could be transferred to face-to-face negotiations.

A class was divided into two groups, and the same two case studies were supplied to all students in both groups. The first group was asked read both cases and give *advice* to the main protagonist in each case. The second group was asked to *compare* both cases and derive an overall principle.

The results showed the "comparison" group were more than three times likely to transfer the principles they derived from the case studies into a face-to-face negotiation situation than those in the "advice" group. Indeed, it was demonstrated that even the quality of advice given did not predict subsequent behaviour plus no-one in the "advice" group drew parallels between the two case studies.

This means that the values of examples are far greater if analogical comparisons among examples are encouraged, but it also shows that there is substantial evidence that suggests groups are very poor at discussing non-common information i.e. a community member's own experience.

This dissertation will attempt to investigate whether an improvement in knowledge transfer can result from decision-making in a virtual environment.

2.3.2 Human-Performance Engineering

Leveraging technology from the visual simulation and virtual reality communities, serious games provide a delivery system for organizational video game instruction and training.

A *serious game* is “a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy and strategic communication objectives” (Zyda, 2005).

These serious games involve pedagogy[1] and activities that educate or instruct, thereby imparting knowledge and skill. MMOGs like Second Life can replicate the types of applications that are described as serious games and could also introduce the idea of *affective computing*, which measures a person’s physical and emotional state, as an extension to the virtual environment it provides. There are an every growing number of “digital natives” who are comfortable with the complexities of complex interaction multi-player applications. This is an opportunity to provide increasing sophisticated representations of knowledge through virtual environments like MMOGs.

Other research has shown that complex multiplayer online games are highly advantageous environments for the acquisition of leadership skills (Byron Reeves, 2007).

The Sloan Leadership Model outlines the following attributes:

- Sensemaking – making sense of ambiguity
- Inventing – turning visions into reality

[1] Pedagogy is the art or science of being a teacher.

- Relating – forming relationships within and across an organisation
- Visioning – creating compelling images of the future

MMOGs promote temporary rather than permanent leadership roles and grant numerous opportunities for leadership practice and the environments are even designed to make leadership easier.

Research has shown that critical leadership features are required when the following exist:

- Virtual economies
- Transparency of metrics
- Connection methods for inter-group communication

Second Life contains all three of these features.

2.4 Training and Modelling with Conceptual 3D/Virtual Reality

Increasing uses, and perhaps the most popular conception, of 3D/VR have been in simulating training with virtual representations of real-world complex machines. The cost effective manner of using VR instead of expensive use of physical objects has resulted in large bodies of research about the best training plans for such devices.

In the design of such systems, determining the training goals early has been shown to result in more efficient design. It is recommended to use VR training for well-versed basic skills, especially in safety areas. Including fault injection is also highly beneficial for obvious reasons
(Larry McMaster, 2000).

An important question that relates to what this dissertation touches upon is if contexts (authentic or visual) are related to knowledge retention, or in other words, if instructional utility of virtual reality-based explorations relates to retention of knowledge.

C. R. Hall (1996) performed an experiment of instructional strategy between 2D and 3D system to examine this question. His findings were that:

- 2D users spent half the time of 3D/VR users with instruction
- 2D users practiced their procedures more
- All 2D users achieved mastery of the experiment while all 3D/VR users either ran out of experiment time or were too exhausted to continue

Another study claimed that "independent of instructional strategies, media does not show significant differences between them for learning" (Regian, 1997).

It would seem that for more complex skills, there may be a substantial advantage afforded by practice strategies that are aided by authentic task representation.

One research piece discovered during the dissertation period was a PhD thesis of a prototype desktop for virtual reality worlds to support navigation of a system during information visualisation and retrieval. What was interesting about this research is that it was modelled as hierarchical data visualisation.

The methods used to evaluate the proposition were search task scoring, subjective questionnaires, navigational activity logging and analysis, and administration of tests for spatial and structure learning abilities (Modjeska, 2000). Overall Modjeska found no performance advantage on spatial layout and in fact discovered that in the fourth and final study, which used a map-like view of the VR representation, the participants in the lowest quartile of spatial ability were significantly lower in their search ability.

Conceptual modelling quality was also investigated in a conference paper (Chen, 2005) that outlined a checklist of properties to ascertain:

1. Understand-ability
2. Expressiveness
3. Processing Independence
4. Check-ability
5. Changeability

Where 1 and 2 would represent *Immersion*, 1 and 5 would combine for *Interaction*, 4 and 5 for *Stereopis* while 3 on its own would lead to animation, evolution and change.

2.5 Research within Second Life

Drew Harry is leading MIT's projects in Second Life. The group presents its work under the *Information Spaces* (Harry) banner and its latest release if the *Agree/Disagree Space*.

In Harry's own words, "We have created a social vocabulary for the space that lets people communicate based not just on text chat, but also by the movement of their avatar around the space. This gives people non-verbal ways to communicate about the meeting. We use familiar architectural cues to make this vocabulary legible. The space is arranged a bit like a sports stadium, with the agree/disagree continuum as the focus. Around that area is space for participants who don't want to commit to a particular position on the continuum. Next to the continuum is also a podium space for a moderator. The moderator has a number of tools at their disposal to manage the discussion. An observational area for people who don't want to participate in the discussion at all is separated by a short wall from the main area. By placing themselves within this space (and relative to other people in the space) participants can communicate their role and attitudes about the topic being discussed.”

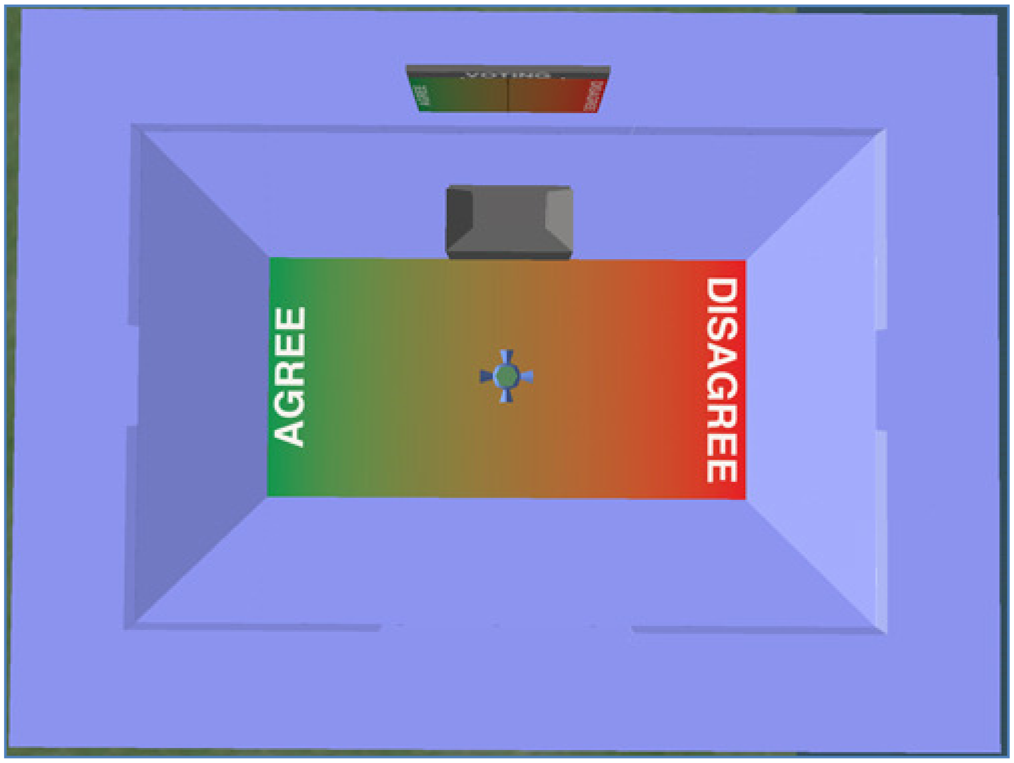


Figure 2: Agree/Disagree Space

Additional tools for this project are a voting indicator and chat and localisation history.

Chapter 3: Design

This chapter outlines the implementation design. Three phases of implementation took place over the period this dissertation was completed. The first phase was the most important and involved design and implementation of the conversion of the policy management system to the Linden Scripting Language (LSL) so that organisational self-management support was available from within Second Life. For this part of the work a Community Based Policy Management System (CBPMS) implementation, using the Ponder policy specification language that deals with access control policy specification, was re-implemented using LSL. The second phase dealt with experimenting with designing and creating 3D representations (objects) from within Second Life that would act as the policy management system that had previously been specified. As the main aim of the dissertation was to create a decision making platform for a community in a multi-use virtual environment to use, the last part of the research was concentrated on creating specific, detailed multi-dimensional decision making aids that show all functionalities of the system.

One of the main factors that affected design process was an incredibly wide potential of the available components and construction options in the Linden Scripting Language and the Second Life client. An example is the fact that almost any discernible three-dimensional basic shape is available to be used and manipulated along any plane or scale. The wide range of settings and possibilities available through Second Life resulting in being forced to adopt a 'trial-and-error' prototyping approach. Exploitation of the main 3D components was the most appropriate method to assess the visual impact and effectiveness of each of the properties tested when designing the decision-making aspects of the platform. The following sections outline the design of the components and constructions of the final system in detail.

3.1 Overview

The specific components of the proposed architecture are divided into five parts: the Second Life Grid, the Second Life Viewer, the Community Based Policy Management System scripts, the decision making interface scripts and the 3D Decision Management Object (DMO). It has already been established that the implementation of the core application will be based on the Second Life virtual environment.

The choice of Second Life as the virtual environment to build the application within was influenced by its current popularity and/or notoriety, its comprehensive application programming interface and its powerful in-world construction and physics. With such an approach, when relying on third-party technology as well established as the Second Life grid, the most of development time and effort can be spent building creative environments and unique content rather than on things like rendering technology, peer-to-peer programming or physical law modelling. With an API and graphical tools provided with such a choice, the resultant research becomes more focused, more flexible and more effective.

The Second Life Viewer is the client and user interface device to interact with the main grid. The default view is of the resident's (or "agent's") own avatar from behind and at a position slightly above.

Because the main area of research of this dissertation is focused on decision making in a 3D virtual environment, it was necessary to use external logic for the policy-based management scheme. However, technical limitations – both software and hardware – resulted in the functionality of the scheme being ported to the Linden Scripting Language so that it was available for interaction within the Grid.

For the decision making aspect of the system, multiple scripts are stored with the policy management scripts to provide rich decision making functionality for the community from with the Second Life environment. These scripts are interacted with through the physical interface of the DMO.

The 3D Decision Management Object (DMO) is the three-dimensional, physical object that acts as an interface to self-management and decision making functionality stored within the object as scripts. As it is a realistically-simulated in-world physical object, it is subject to be manipulated through a variety of actions and communication techniques. This means that actions such as touch, movement, force and gestures as well as contacts such as textual chat messages and notecard-style objects are available as possible methods of interaction and information passing.

The functionality offered by the core LSL API provided by Linden Labs must also be mentioned. The depth and sophisticated of the API, allied with the range of services it outlines, meant that the entire development of the system would not have been possible without it.

A detailed technical discussion of these components as they were implemented is undertaken in the next chapter.

3.2 Platform Process

In order to conduct a process of simulation, a user is required to perform certain steps. However, as the platform is contained within a multi-use virtual environment, there is significantly more freedom available to the user to in terms of where and how they interact with the environment through their avatar. Therefore for this illustration it is assumed that the user is performing within a prescribed set of actions.

The start of the operation depends if the user has initialised the system by instructing their avatar to “touch” the DMO. Once the user has completed this step, the system enters its “ready” state and is accessible to interaction through a myriad of actions and communications. The user now needs to perform recognised instructions for the DMO to function. The full list of available actions and communication techniques will be outlined in the *Implementation* chapter; however they include touch, textual chat, gestures, physical movement, object manipulation and object passing.

If the performed instruction conforms to the required attributes of the desired task (i.e. if a chat message is used, it must be in the correct grammatical format; if a movement is used it must be on the surface of the DMO and within the specified boundary), the DMO will check to ensure that all prerequisite settings (for example, the community structure) are in order. Once that has been confirmed, the desired action will be carried out. Should any response or return message exist, it will be displayed and/or be transferred to the appropriate user. When these steps have been completed the DMO will return to its “ready” state.

If the performed instruction does not confirm to the mandatory characteristics of a successful request, the DMO will transmit a refusal message and return to the ready state.

The entire process with more detailed information about implementation and activities is described in the *Implementation* chapter.

3.3 Functionality Requirements

The main goal for this project is to provide a platform that permits an online virtual community to collaborate in 3D to self-manage their access policies and to reach decisions. This goal will allow communities to resolve conflicts and form clear views of how they organised themselves and reached decentralised communal decisions. Previous chapters give a broad overview, in terms of mental models, 3D learning, transfer of tacit knowledge from an individual to a community in a virtual environment, of the main key objectives for the platform which in the initial stages of the project were outlined in the most significant terms.

3.4 Conclusion

In conclusion, development of the entire system will involve the coding of two separate component scripts and a design and construction of a three-dimensional object that acts as the intelligent interface to the above scripts' functionalities:

- a lightweight Community Based Policy Management System written in the Linden Scripting Language providing a Second Life community the ability to define itself to reflect the natural structure of the group.

- Cognitive Decision Making functions that attempts to permit a community in Second Life to come to decisions using a far greater number of communication methods and social vocabulary by incorporating spatial and intentional information.

- a virtual environment construct, used to transfer tacit knowledge in an easier manner, to share in a higher level of knowledge representation and richness of perspective and to enhance interaction leading to increased internalisation and reduced equivocality.

Due to the nature of the system and the components that are used throughout development, at this phase of the research it is difficult to present more specificity and detail about the proposed implementation. The investigation into how a virtual environment will affect how a community makes decisions on behalf of itself needs more practical investigation.

As a result, the design process somewhat overlaps the implementation process and vice-versa. The limitations of the third-party engine anticipated, and indeed limitations with other external sources already experienced, could affect overall assumptions made at the beginning of the research. Also, the nature of virtual environments opens the area of unexpected possibilities for design at this stage.

Chapter 4: Implementation

This chapter details how the design ideas and principles outlined in the previous chapter were implemented. The first section reviews the supporting tools and technologies used for the implementation. The remaining sections introduce how the actual system and its subcomponents were constructed.

4.1 Technology Review

Of the tools and technologies mentioned in the *State of the Art* chapter, several were used for implementation. Due to the completely different nature and character of the three subcomponents developed, a separation of the implementation description into three separate parts is required.

4.1.1 The Linden Scripting Language

The Linden Scripting Language (LSL) is a simple, powerful language that is used within Second Life *scripts* to attach behaviors to the objects found in SL. It follows the familiar syntax of a C/Java style language, with an implicit state machine for every script.

LSL is a state-event driven scripting language. A script consists of *variables*, *functions*, and one or more named *states*. Each state contains a description of how to react to events which occur while the program is within that state.

The system sends events to the script. These events encompass such things as timers, movement, chat and collisions (with objects in the virtual world). Scripts can change most aspects of the state of the object and communicate with other objects and *agents*. As soon as a script is added to an object, it begins to execute.

Multiple scripts may also be attached to the same object, allowing a style of small, single-function scripts to evolve. This leads to scripts that perform specific functions ("hover", "follow", etc.) and allows them to be combined to form new behaviors.

The text of the script is compiled into an executable byte code, much like Java, as LSL is a strongly typed language. This byte code is then run within a virtual machine on the simulator within a Linden Lab server. Each script receives a time slice of the total simulator time allocated to scripts, so a simulator with many scripts would allow each individual script less time rather than degrading its own performance. In addition, each script executes within its own chunk of memory, preventing scripts from writing into protected simulator memory or into other scripts, making it much harder for scripts to crash the simulator.

The LSL API provides over 300 library functions and LSL developers can also define additional functions. LSL's native data structures include integers, floats, strings and vectors (3D co-ordinates. There are also heterogeneous lists.

There is no built-in persistent data storage. On the other hand, scripts continue to run even when a user is not logged in, and if an object is saved to an agent’s inventory, and then re-introduced into the world later, it still maintains its state.

Some functions in LSL have built-in delays, which range from a 0.2-second delay when moving a non-physical object to a 20-second pause when sending an e-mail message. The delays are intended to prevent developers from writing LSL scripts that could overtax system resources. Memory available to LSL scripts is capped at 16 KB (both heap and stack), which places a practical limit on how much a single script can do.

4.1.1.1 Experience with implementing with LSL

Most of the existing documentation on LSL is written from a novice perspective due to the case that many SL citizens are first-time programmers. From the view of an experienced software developer, the language seems to present so many restrictions and limitations of the Second Life scripting engine that a reasonably complex project can quickly encounter serious difficulties.

Objects (in the software development sense) do not exist in LSL, just basic primitive data types. The only collection-like data structures are *lists*, which are strictly one-dimensional and cannot be modified in place. This dissertation's attempt at programming a multidimensional collection was largely successful although it is recognised that list manipulation in Second Life leaks memory. With such a limited amount of memory to work with in the first place, sophisticated data management can become very frustrating.

Scripting across objects and between multiple scripts is also difficult as the functions that allow communication between them are quite simplistic. There is no concept of *test-driven development*[2]. There is also no concept of long-term persistence in Second Life as scripts given to other objects can be reset.

For the development environment, the LSL-Editor, a standalone LSL script editor and run-time environment for Windows, was chosen. The LSL-Editor has the ability to compile and execute LSL scripts off the Second Life Grid. This was deemed superior to the ASCII Content Editor available through the Second Life Viewer as it was quicker at compiling, had code formatting tools and did not require the Grid to be online.

[2] Test-Driven Development is a software development technique that involves repeatedly first writing a test case and then implementing only the code necessary to pass the test.

4.1.2 Community Based Policy Management System implemented in LSL

It has already been established that in order to investigate how decisions are made by a community in a virtual environment it was determined that by introducing a system that allows a community to manage its internal structure and grouping abstractions used in authoring policies, it would provide a series of opportunities to evaluate how the community make decisions to organise itself.

Therefore an external CBPM-type system needed to be used. This CBPM would use “the notion of a *community* as the grouping abstraction with the aim of allowing groups within the organisation itself to define communities to reflect the natural structure of the organisation” (Kevin Feeney, 2006). As communities in Second Life can potentially exist with the same hierarchy of authority as in real life, the ability to represent organisation structure using three-dimensional tools within a virtual environment would be highly beneficial.

The implementation of the CBPM System is structured, with the components that are utilised within this dissertation highlighted, as in **figure 3**.

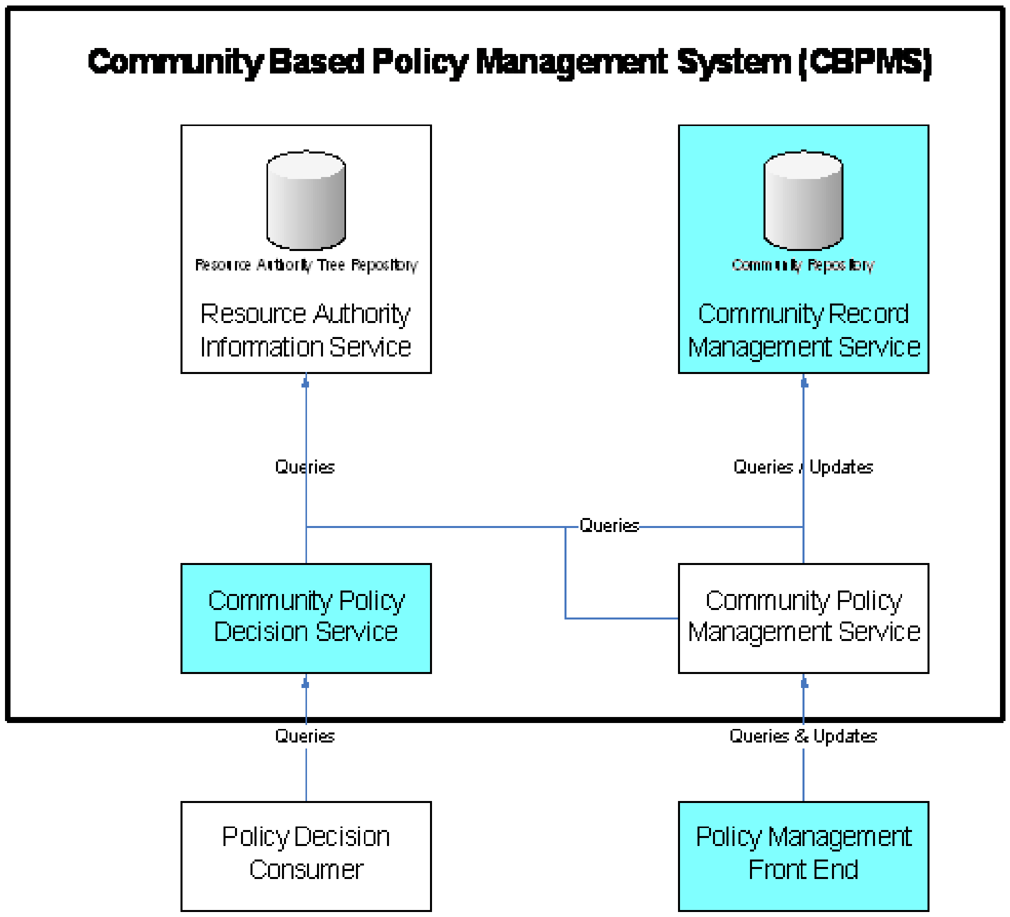

Community Based Policy Management System (CBPMS)
Resource Authority Tree Repository
Resource Authority Information Service
Community Repository
Community Record Management Service
Queries
Queries / Updates
Queries
Community Policy Decision Service
Community Policy Management Service
Queries
Queries & Updates
Policy Decision Consumer
Policy Management Front End

Figure 3: CBPM System Overview

4.1.2.1 Limitations of external CBPMS

The dissertation's original aim was to utilise the full external implementation of the CBPMS developed by Feeney as described in (Kevin Feeney, 2006). In this instance, it was envisioned that policy management functionality would have been accessible from within the Second Life environment by utilising XML-RPC[3] calls to the service. However, a number of limitations and difficulties were encountered:

- Scaling issues with the Second Life grid architecture; at the time of writing, Linden Labs had in fact only a single front-end server that handled XML-RPC communication. This meant that messages were queued and, at an attempt of load-balancing, there was a three-second delay imposed between processing them, and hence, receiving responses.

- Second Life's XML-RPC support is still at it early stages. With such limited architecture dedicated to the technology already outlined above, Linden Labs has also placed a limit to the size of incoming messages; the current threshold is 254 bytes. This limitation did not suit the external CBPMS implementation which requires large strings to hold all appropriate information about the community structure and/or policies.

- Linden Labs has also forbidden bi-directional XML-RPC requests. Currently only outbound XML-RPC requests (i.e. from the grid to an external architecture) are permitted. While in this scenario it would have suited to contact the system from in-world, the response messages would have had to have been intercepted by a proxy service that was listening out for responses from the XML-RPC based CBPMS to Second Life. This service would then have to parse and reformat the message (due to the message size limitations outlined above) and then queue before attempting to push the message using HTTP requests to the SL Grid. With the dissertation's main focus being the evaluation of visual decision-making, it was deemed that this solution required a disproportionate level of effort.

[3] XML-RPC is a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism.

- Finally, the implementation of the CBPMS is located within the Trinity College network. As Second Life does not offer the option of routing through a named proxy, with initial attempts to configure the network proving difficult to conform to the dissertation schedule, allied with the above issues of the communicating with the CBPMS, this approach was discontinued.

4.1.2.2 Overview of implementation of existing CBPMS

As can be seen from **figure 3**, an implementation of a CBPM System contains a broad array of services are constructed to offer the community the necessary levels of flexibility, dynamism, decentralisation and autonomy in management.

For this dissertation's implementation, focus was on how the primitives offered by the Community Record Management Service (CRMS) could be replicated within Second Life, so that it can be used to enable self-management from within the virtual community. The external implementation would have interacted with the CRMS through the Policy Management Front End tool which consisted of web-based forms. However, as the proposed implementation was to be based in a 3D virtual environment, this approach would have to completely different and the methods used will be outlined later in this chapter. The implementation would also involve simulating the use of the Policy Reasoning Service (PRS) which operates within the Community Policy Decision Service (CPDS). The CPDS evaluates policy decision requests from the policy decision consumer.

4.1.2.3 Conversion of the Community Record Management Service

The CBPMS Community Record Management Service (CRMS) supports 12 management primitives. Through a dynamic progression process, these primitives permit an organisation to define its policies.

Every resource that is managed is modelled as an authority tree, which is divided into an action tree and a target tree. These *Resource Authorities* provide an authority scope for each community. By delegating this authority down the community hierarchy, the community can specify what events the internal sub-community can author policies for.

The CRMS primitives are outlined in below:

genesis/ expel	Create / Destroy a root community
spawn/ cull	Create / Destroy a sub-community of an existing community
delegate/ recall	Delegate authority to a sub-community or recall a delegation
policy/ revoke	Specify a policy for a community or remove specification of an existing policy
federate/ withdraw	Join or leave a federation
grant	Assign ownership of a resource to a community
gatekeeper	Define a membership rule for a community

4.1.2.4 Outlining the Policy Specification Language and Evaluation Engine details

In order for this proposed implementation to be operational, the previous implementation's policy specification language and policy evaluation engine chosen must be examined. The subset of the Ponder policy specification language that deals with access control policy specification was selected. The reasoning behind this selection was Ponder, like many policy specification languages, is integrated into a service which implements the basic Policy Reasoning Service specification that specifies access control policies. Ponder is also expressive and uses a well known syntax.

The challenge is this implementation is to attempt to utilise the Linden Scripting Language so that it can perform like a policy specification language and a policy evaluation engine. For example, in the previous implementation, policy specifications take the following form:

```
     inst ( auth+ | auth- ) policyName { [when constraint-
Expression ; ]}
```

The above policy specification has the following semantics: the constraint-Expression is evaluated by the policy engine (meaning the specified policy exists) and if it evaluates to true then the return value is either auth+ or auth- (meaning authorisation has been granted or declined respectively), depending on which is specified.

Another important point to note as regards how the implementation should be converted to LSL is the assumption that a mechanism exists for context information sent to the Policy Reasoning Service (the engine) to be translated into a form that LSL can understand. In the circumstance of a 3D virtual environment where a wide variety of non-verbal communications are used, this is particularly important. It is also assumed that whatever information is required in order to evaluate the constraint expression is included in the context.

4.1.2.5 Transformation of the Resource Authority Tree

The final section to articulate before outlining the converted implementation is to describe the concept of the resource model. As there is an assumption in the initial stages of a community managing itself through a CBPMS that there are no resources managed by the system and that there are no policies within it (perhaps other than basic membership rules), a resource model is built by constructing a target tree and an action tree.

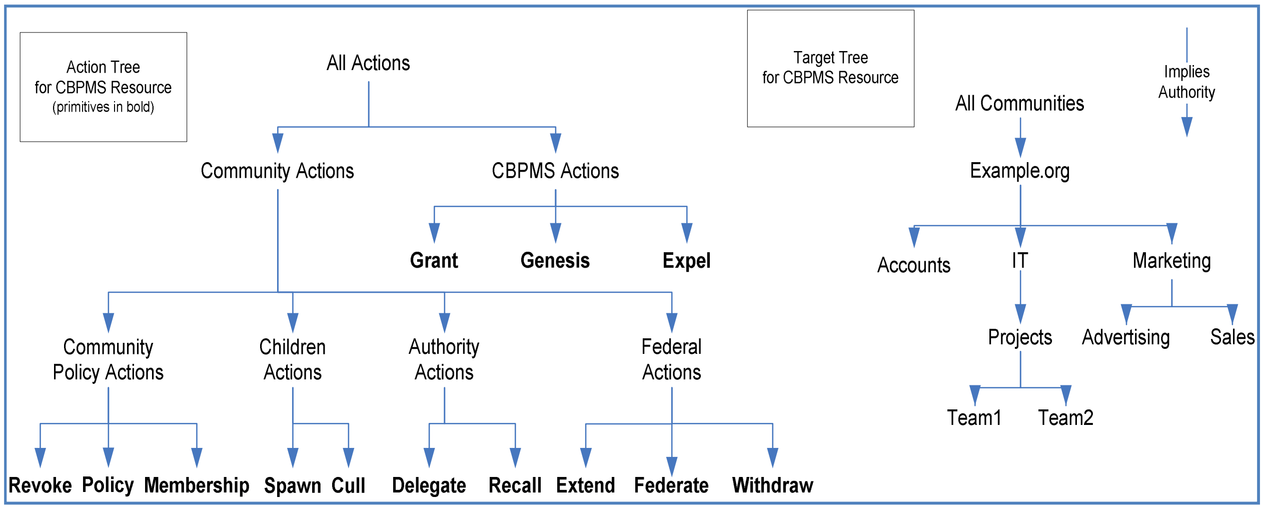


Figure 4: Specification of CBPMS Resource Tree for Self-Management

These trees represent the resource types and the actions that can be performed upon them. In fact, every resource that is managed is modelled as an authority tree, which is divided into an action tree and a target tree. A resource authority “is a triple [R, T, A]” where:

- R is the resource model
- T is a node on the resource's target tree
- A is a node on the resources action tree

Now that the structure of the policy-based management scheme has been addressed, the LSL-based implementation solution can be shown.

Firstly, the nature of the resource tree will be addressed. In this instance only the Action Tree will need to be replicated within Second Life as the Target Tree will in fact be represented by 3D objects and resident's avatars.

In the above tree diagram, the *All Communities* will be represented by the final 3D structure (the DMO) of the CBPMS represented within Second Life. The *root*, in the above figure called Example.org, will be represented by the designated owner of the DMO and therefore head of the community. The explanation of how the Target Tree will be dynamically and spatially created will be detailed later in this chapter.

4.1.2.6 Using LSL for Action Tree, CRMS primitives and Policy Lists

As had already been established, the Linden Scripting Language, and indeed Second Life itself as a development environment, has seriously limitations. One of the limitations that were identified, the lack of sophisticated storage constructs like arrays (either two-dimensional or multidimensional), linked lists or hash-maps, resulted in serious design and implementation difficulties. Hierarchical structures like trees and policy tables are, in essence, basic persistence and collection solutions in memory.

Therefore it was necessary to create a multidimensional collection in LSL by using the basic functions and primitives available within the language.

To resolve this issue, some ‘abstract thinking’ was required. As the basic list primitive in LSL can only hold strings and integers (not other lists), the challenge of creating a multiple dimension collection was difficult. In the end a series of functions were written than used basic lists to manage the array structure, the indices and the data itself. These functions managed the structure by using LSL’s own `list` construct to manage the array itself, the array’s indices, and the array’s data.

It would store this data in one large list by compressing the data into character strings separated by randomly generated string separators. When navigating to desired elements the function would recursively call itself, decompress any available structure and use the decompressed index values to select the correct

Using this solution, the components of the Community Based Policy Management System can be constructed.

For the Action Tree, each node was designated an identifying index value. The hierarchy of this CBPMS resource would be represented within the method calls of the scripts itself. Each of these methods would correspond to the execution of a primitive upon that non-primitive node. Each of these methods would take four parameters:

• `key subject`	(Subject)
• `key resmodel`	(Resource Model)
• `key targetnode`	(Target Node)
• `key actionnode`	(Action Node)

As can be seen, the last three parameters listed above correspond to the resource authority triple [R, T, A].

The subject refers to the destination of the policy authorisation request – in this case it would be the avatar in Second Life that the request is being acted upon. The resource model is the managed resource that the subject is attempting to interact with through one of the CRMS primitives. The target node is the member or grouping of the community that the subject is attempting to gain authority over. The action node represents the action that the subject is requesting authority to perform. The type of the each of the parameters is 'key' which is an identifier that is used to distinguish all actions, targets, subjects and resources within and available to the community. Each of these constituent keys of the application will be stored in type `list` variable.

This explanation is perhaps clarified with the use of an example. In this scenario, ultimate authority is to be transferred to the *root* by using the **grant** primitive. This means that *root* (using the diagram with example hierarchy already illustrated, called Example.org) with be granted authority over the resource authority that is contained with the entire '*CBPMS'* resource, the '*All Actions'* action and the *'All Communities'* target.

Firstly, the `list` variable, which is responsible for tracking who has been granted what authority, is declared:

```
list grantAuthKeys = [];
```

Then the Policy Reasoning Service will identify the appropriate 'set Action function' to call based on the key of the `actionnode` requested and supply it with the correct Ids that correspond to the subject, resource model, target node and, of course, the action node.

For example, the `All Actions` node on the Action Tree corresponds to the identifier of 1. Therefore the function that the PRS has linked to this identifier is called, in this case:

```
setAllActsAuth(
  key subject, key resmodel, key targetnode, key
actionnode) {
    setCommActsAuth(subject, resmodel, targetnode,
actionnode);
    setCBPMSActsAuth(subject, resmodel, targetnode,
actionnode);
}
```

It can be now observed how the hierarchy of the Action Tree is replicated. As the `Community Actions` and `CBPMS Actions` are children of the `All Actions` node, the child functions are called and passed the same parameters. These functions would, in turn, call their children's set action functions:

```
setCBPMSActsAuth(
  key subject, key resmodel, key targetnode, key
actionnode) {
    setGrantAuth(subject, resmodel, targetnode,
actionnode);
    setGenesis(subject, resmodel, targetnode, actionnode);
    setExpel(subject, resmodel, targetnode, actionnode);
}
```

Concentrating on the **grant** primitive alone, the following method shows how the customised multidimensional array functions are utilised to replicate the structure of the CBPMS resource authorities:

```
setGrantAuth(
 key subject, key resmodel, key targetnode, key actionnode)
{
  grantAuthKeys = setArray(grantAuthKeys, [0, 0],
[subject]);
```

```
  grantAuthKeys = setArray(grantAuthKeys, [0, 1],
[resmodel]);
  grantAuthKeys = setArray(grantAuthKeys, [0, 2],
[targetnode]);
  grantAuthKeys = setArray(grantAuthKeys, [0, 3],
[actionnode]);
}
```

The initial `grantAuthKeys` list has now been transformed into a four-column array where the key is the identifier of the subject, and has corresponding identifier values to signify the what authority he/she was granted, on what community sections and on what resource.

Now, when decisions are to be made about whether a subject (in Second Life this would be an avatar that either represents him or herself) individual or whether it is being used to symbolise a community, the appropriate multidimensional collections are traversed. Note that recursive traversing is used because subject's parent Ids are stored within a list. This allows the traversal of the simulated tree structure, however at some programmatic cost of speed and memory. By simply having an entry within one of these collections means a simple method for identifying which subjects have authority, and of what type, over the community's resources. These collections can then be searched to return the equivalent of `auth+` and `auth-`, showing how the main responsibilities of the original CBPM System have been replicated.

Similarly, policy specification methodology and management can be simulated. Consider the following scenario; the *root* community has been granted the CBPMS authority. This means that access to the CBPMS can be managed by the system itself. Therefore applications wishing to access the CBPMS primitives can generate appropriate policy decision requests and either allow or deny access according to the semantics of the policy decisions that the CBPMS will return. However, since the CBPMS resource itself is part of this scheme and there are no policies in existence for the resource, a bootstrapping problem exists.

The solution is to create a *default* policy in the root community. In this case we will simply call this policy ‘initial’ and it is assumed that the job of setting up a self-managed community structure has been allocated to the appropriate individual within that community. This is accomplished as follows:

Firstly, two `list` variables, which are responsible for tracking created policies and tracing who has access to those policies, are declared:

```
list policies = [];
list policy = [];
```

The former is a two-dimensional array that associates a policy ID with a named policy function. Unfortunately, it was impossible to create functionality that permitted dynamic specification of policies within the dissertation timeframe; therefore the available policies in the system have been pre-determined.

The latter is similar to our previous `grantAuthKeys` list that tracked authorisations. In this instance, this array will be used to connect subjects to policy functions identified through the `policies` array. For example:

```
setPolicy(
  key subject, key resmodel, key targetnode, key
actionnode, key policykey) {
   policy = setArray(policy, [0, 0], [subject]);
   policy = setArray(policy, [0, 1], [resmodel]);
   policy = setArray(policy, [0, 2], [targetnode]);
   policy = setArray(policy, [0, 3], [actionnode]);
   policy = setArray(policy, [0, 4], [policykey]);
}
```

The only difference here to the previous ‘set Action’ function is the inclusion of a fifth column that uses the key value to represent the policy to be associated with. Once the policy has been associated, control is redirected to the method that details the policy, along with testing constraint expressions and returning an integer value 1 or 0, representing the Boolean true or false i.e.

```
integer canInvoke(string parameterdata) {
   // test parameterdata (if exists) verses constraint
   …
    // lookup policy table to find out if policy can be
invoked
   …
    // return response
   …
}
```

Using this implementation solution, many features that are available in the external full PHP-based implementation of the CBPMS have been sufficiently replicated for the purposes of this dissertation. How this implementation is combined with cognitive decision making functionality accessible through a three-dimensional object is discussed as this chapter continues.

4.1.3 3D Decision Management Object

Before continuing the description of functionality implemented within LSL scripts, the physical object that acts as an interface to self-management and decision making functionality stored within the object as scripts is introduced.

The main goal of the entire system is the visualisation of the decision-making process that a community undertakes when self-managing its structure and resolving policy conflicts. Therefore it is an essential part of the research and development that the actual visual effect that will serve this function be representative of the main characteristics that would contribute to the evaluation of this research. In order to develop this kind of visual facet, the following aspects need to be taken into consideration:

i. the chosen three-dimensional object needs to be conceptually sound to associate its provided functionality with a physical representation of such an object being able to supply these abilities. However, it must be noted that the entity must not be as abstract as to its purpose to be judged unfathomable.

ii. the object will have to be as dynamic as possible, having the ability to respond to a wide variety of communication techniques, both physical and non-physical.

iii. the selected object will have to be of suitable dimensions for the evaluations that will utilise it while at the same time be unobtrusive for general virtual environment representation.

iv. as Second Life is a truly three-dimensional virtual environment with the ability for avatars to move easily in any direction or plane, the chosen design will need to consider all of the following attributes:

   a. **Spatial**
      aware of location, direction, area and space.

   b. **Temporal**
      understand the concept of time and how it will affect its interactions with other objects.

   c. **Communication**

be adept at communicating itself to others in the virtual environment.

d. **Intent**
anticipate actions and display such expectancy.

e. **Influence**
permit a series of influences (e.g. physical forces, gestures) to be included as influences on how the object behaves.

f. **Clarity**
able to express itself clearly and succinctly.

g. **Ergonometric**
to be, as much as possible, designed towards ease-of-use for the interacting agent.

Nearly everything in Second Life is made of *prims*, short for primitives. These are the building blocks of SL and include the basic shapes and objects made available through the *Building* tools of the Second Life Viewer.

To be able to build, the potential construction needs to be in a place where building is allowed. There are public sandboxes which allow people who don't own any land to build, however there is a *time-to-deletion* limit on each primitive in these areas and therefore is not suitable. Hence, a 512m^2 parcel of land was purchased for this dissertation’s research.

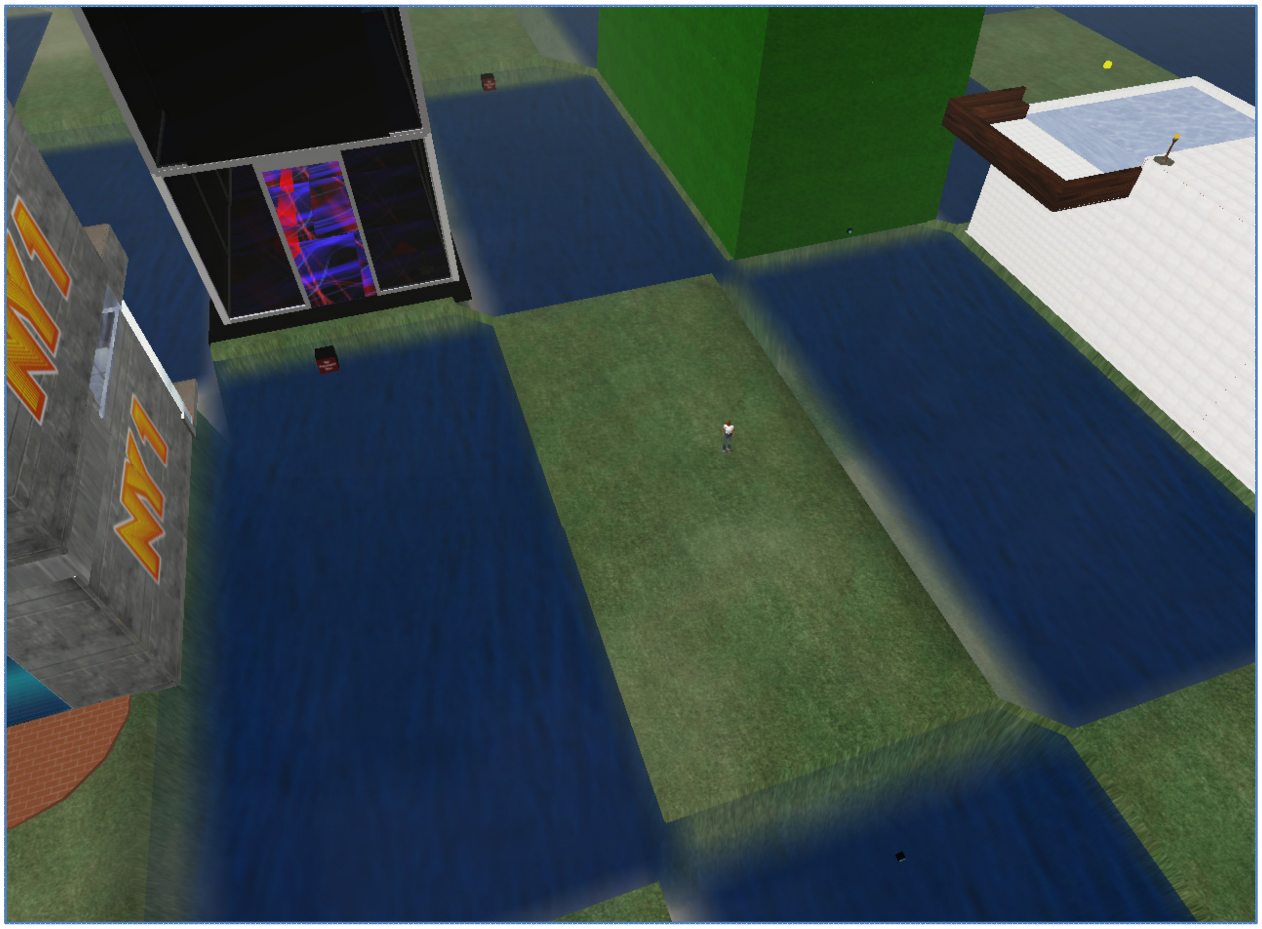


Figure 5: the purchased 512m^2 parcel with avatar

Each parcel of land in SL has a prim limit, and the prim limits are related to the size of the land. Thus, in this case, as a first-land-sized piece of property of 512 square metres, it is permitted to use up to 117 *prims*. A simulation-sized island of land allows you to use up to 15,000 *prims*. For this implementation is was determined to minimize any unnecessary usage of *prims* on the parcel of land.

There were also API considerations in the Linden Scripting Language that affected the choice of design. Sensor events in LSL send out pulses of a specified diameter three hundred and sixty degrees around a central point. This, of course, limits the external shape of the object.

Another consideration is that multiple avatars will be interacting with the object at the same time. There exists a mechanism in Second Life to *touch* an object – *touch* means you are able to affect an item without being physically near to it. Touching does not require immediate proximity, instead a line of white light acts as proxy and mediator to actual touch. However, objects can only be touched by one other object, of which an avatar is one. Therefore, alternative communication techniques, for instance, an avatar's physical location and kinetics must be factored into the design. This means that perhaps a large, wide and flat surface would be best suited.

After all above stipulations were considered, the following design was chosen:

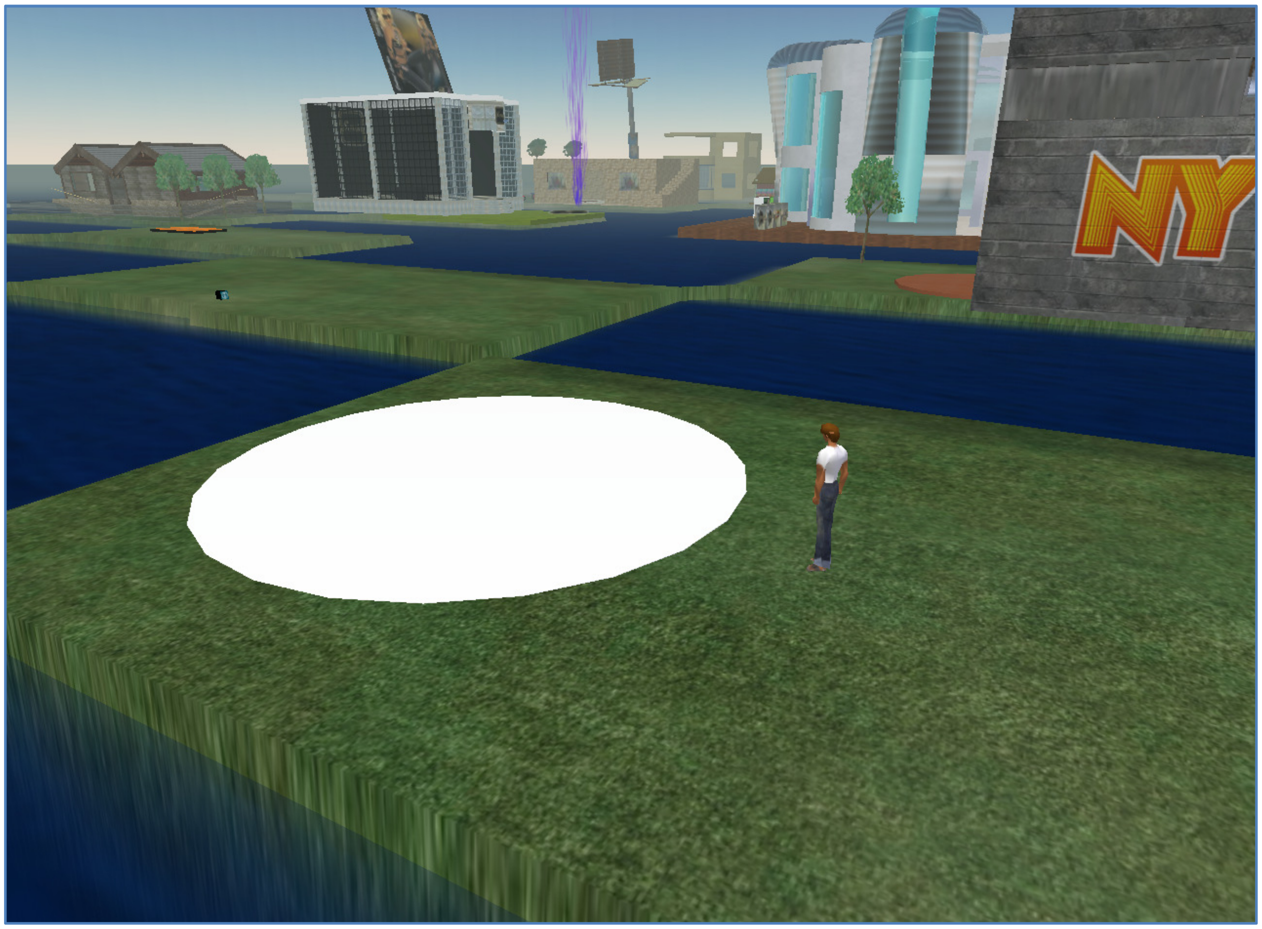


Figure 6: the chosen design

The circular shape was chosen to maximise the benefit of using the `sensor` state-event trigger described above.

The radius of the circle was selected to be the maximum 5m.

The shape is in fact a *can* three-dimensional shape; however the height of the object is only 10cm. This allows avatars to walk on top of the surface of the object and therefore within the sensor boundaries of the object.

The colour of the circle is white to maximize visual clarity of an avatar's spatial and communicative information.

4.1.4 Cognitive Decision-Making functions

In this dissertation, a community in a virtual environment attempt to make decisions with the goal of self-organising their community structure and management. As the community was represented by a group of avatars, which are in turn controlled by human agents, the decision-making implementation would have to be approached from the perspective that it was Cognitive Decision-Making that was taking place. This means that a much more sophisticated level of conscious choice selection compared to an alternative form-based

Cognitive Decision-Making (CDM) has been described as "a human-like complex mental process; the essential aspect of high intelligence". More formally, it is "a mental or computational activity implied by a necessity of a choice without yet either known *criteria* or known *alternatives* or with insufficient information."

As decision-making among a community of humans (albeit represented in a virtual manner) is being investigated, the Top-down Object-Based Goal-oriented Approach (TOGA[4]) to general cognitive decision-making can be modelled.

This generic decision-making model is based on three fundamental concepts (IPK):

- Information
- Preferences
- Knowledge

4.1.4.1 Basic Decision-Making Ontology[5]

Now that the decision-making approach has been identified as a cognitive process, the required definitions can be defined:

- **Choice** - a mental/computational process which relies on the application of preference rules. It can be either reasoning-oriented or action-oriented.

- **Criteria** – a set of preference rules applicable to the choice.

- **Alternatives –** a set of candidate abstract entities/events to be chosen.

[4] TOGA is the goal-oriented knowledge ordering (conceptual modelling) tool for the specification and system/process identification of real-world complex problems.
[5] An Ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts.

With this ontology now defined, a subset of possible actions and communication techniques used within Second Life can be identified as potential decision-making processes that display a cognitive element.

The initial set will be populated by the nature of the architecture, functions and interactivity of Second Life itself. What this means is taking advantage of the pre-existing source supplied by Linden Labs. Therefore the state-event driven methodology of LSL (in the sense of a finite state machine) means we can deliver alternatives of interactivity methods e.g. touch, sensors, listeners to communication (either verbal (chat) or non-verbal (gestures)), temporal events based on timers etc.

The very nature of the virtual environment where the source of activity is directed by avatars and objects that are either controlled by or specified by human agents means that choice exists in what decisions are made, simply by associating those decisions-making points with reasoning dictated by the human user of the virtual environment. When a user performs a specified action on an object which has decision-making capabilities (i.e. the 3D DMO), that action can be linked with a particular decision-making (or indeed conflict resolution) function.

Also, action-oriented cognitive decision making can be implemented that satisfies the defined ontology. For instance, voting can now be expressed in new manner where an avatar spatial representation can define his or her human agent's intent. The *choice* would be the motion put forward; the *criteria* would be the located within the stated boundaries of the voting area; the *alternatives* would be the voting value itself, expressed through the avatars current position.

The dynamism of the decision-making process becomes evident as participants can interact with each other on various levels (for example, physically (through position or gestures), verbally and textually) and also introduce the concepts of influence and visual representation of mental models. For this latter point, consider the following example; when an avatar chats during a voting session, a 3D object – in this case a ball – is *rezzed*[6], floats upwards and through its visual representation can demonstrate a variety of information. This information would include what was said and by whom (intent), the current opinion of that source on the motion (e.g. red signifying opposition) and the temporal nature of the expression (the closer it is to the voting floor, the more recent it was).

The object being acted upon can also give real-time feedback to the community about the current state of their decision-making and how it is managing itself through the decisions made by interacting with the self-management functionality provided i.e. the policy-based management scheme.

So, as will be outlined in our evaluation, this methodology can allow a community to have potentially complete freedom in how it expresses its decision

4.2 Conclusion

In this chapter, the implementation of the reflection decision-making platform has been outlined and described. A technological review provided a discussion about the development environment and language used and how this affected the implementation. Each component that had been outlined in the design, starting with the CBPMS converted to LSL to provide in-world self-management within Second Life, followed by the three-dimensional object that provided the base for associations between decision-making functionality and self-management functionality. Finally, the nature of the decision-making process implemented was discussed and presented how all three components could be combined to function as a developed decision-making platform for a community in a virtual environment to self-manage.

[6] Rezzed is a Second Life term for an object to appear; to be given resolution.

Chapter 5: Evaluation

This chapter presents results, evaluation and a discussion of the developed platform. Although the system consists of three main components, the evaluation will be based on an incorporation of all three, as it is only when they are combined that the dissertation goals can be examined and assessed.

5.1 Hardware used

The development, implementation and evaluation of the system were conducted on three separate, and quite diverse, computers with the following configurations:

#1: General Information: Dell Dimension 9150	
Operating System	Microsoft Windows XP Professional with Service Pack 2
Processor	DualCore Intel Pentium D 830, 3000 MHz (15 x 200)
Memory	2048 MB (DDR2-533 DDR2 SDRAM)
DirectX Version	4.09.00.0904 (DirectX 9.0c)
Display Devices	
Graphics Card	nVIDIA GeForce 7800 GTX
Display Memory	256 MB
Resolution	1280 x 1024
Others:	
LSL-Editor v2.14	
Second Life Viewer v1.18.1(2)	

#2: General Information: Targa Traveller 811	
Operating System	Microsoft Windows XP Professional with Service Pack 2
Processor	Mobile AMD Athlon 64, 1800 MHz (9 x 200) 3000+
Memory	512 MB (PC2700 DDR SDRAM)
DirectX Version	4.09.00.0904 (DirectX 9.0c)
Display Devices	
Graphics Card	ATI MOBILITY RADEON 9600/9700 Series
Display Memory	128 MB
Resolution	1280 x 800
Others:	
LSL-Editor v2.14	
Second Life Viewer v1.18.1(2)	

#3 & #4: General Information: Dell Latitude D400	
Operating System	Microsoft Windows XP Professional with Service Pack 2
Processor	Mobile Intel Pentium M 725, 1600 MHz (16 x 100)
Memory	512 MB (PC2700 DDR SDRAM)
DirectX Version	4.09.00.0904 (DirectX 9.0c)
Display Devices	
Graphics Card	Intel 82852/82855 GM/GME Graphics Controller
Display Memory	64 MB
Resolution	1024 x 768
Others:	
Second Life Viewer v1.18.1(2)	

It must be noted that save the Dell Dimension 9150 with had a 256 MB Graphics Card and 2 Gigabytes of memory, the hardware used had difficulty achieving a level of visual smoothness when the Second Life Viewer was running. Although all contributors in the evaluation affected by this reported that it did not prevent them from participating in the evaluation, they did report that this did affect their evaluation is a negative manner.

5.2 Evaluation approach

In the initial stages of this dissertation, it was anticipated that as the subject matter was examining a new area and was somewhat proof of concept, the undertaking of collecting community metrics was considered spurious as the results would lack anything of similar nature to compare against. It was envisioned, and indeed had been simultaneously in progress during implementation, that the author would perform a single evaluation of the overall functionality, visualisation impact and usability of the system without other participants. The results would be subjective opinion in nature and the experiences encountered would be contrasted with what were deemed to be required levels of the said evaluated concerns.

However, as the complexity of the system grew with the increasing functionalities being provided, it was determined that a more concrete evaluation should be performed.

This evaluation focussed on analysing and evaluating the effectiveness of using three-dimensional virtual environments to capture tacit knowledge in a community so that it can be used for cognitive decisions to be made in order control the community's structure and policies.

5.3 Evaluation outline

The evaluation approach chosen was a series of four scenarios that were design and implemented, using the platform outlined in the previous chapters, beforehand and would form a solution to a general problem. The problem selected was a subset of the example problem proposed by Feeney, Lewis and Wade in *Self-Managing Organisations*. A comparison would not be made however between how the solutions to the respective problems as that approach would only capture a fraction of the evaluation goal of being assessed.

Instead, feedback from the participants was to be received so that an evaluation of the effectiveness of the implemented system in the virtual environment at presenting the platform for reflective decision-making (as outlined in the previous chapters) could be examined. The questions would be asked after each scenario had been completed.

For the feedback received per section, the general theme of all scenarios was evaluated at the end of the evaluation by the author. The feedback sections of interest were:

- **Spatial**
  Evaluating the participants' ability to judge scale and spatial representations

- **Cognition**
  Ascertaining the information internalised by the participants during the scenario

- **Immersion**

Determining the level of 'reality' [7]the client experienced with the system

- **Richness of Visuals**
  Establishing the visual effect of the scenario had on the client

- **Interactivity**
  Examine if the client felt about interacting with the scenario

The above sections were selected because the author deemed them to be the most appropriate for evaluating the range of functionalities offered by an experimental system that had been outlined throughout the *Implementation* chapter.

There were four participants in the evaluation. They were each assigned a number and then a computer (specifications detailed above). The participants were also given existing Second Life accounts and some basic training in using the SL Viewer. The author's Second Life account and avatar was used in the evaluation but by one of the participants; the author organised and observed the evaluation only.

Three of the participants were first-time Second Life users, while the other had revealed that they had "messed around with it a couple of times". All participants could therefore be considered as novice SL users and therefore were in a position to accept the abstract nature of the visual representations easily, rather than have to re-educate experienced Second Life users.

[7] Reality is the essence of their presence in the environment.

5.3.1 Evaluation Initialisation

The problem introduced is to use the 3D DMO, which has been implemented as a self-management interface for a CBPMS system, along with cognitive decision-making tools also incorporated within the DMO, to configure a desired final community structure as shown in **figure 7**:

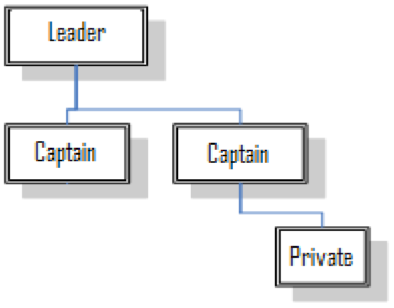


Figure 7: Community Structure

In order for the participants to complete each scenario, they must use a prescribed list of actions and communication techniques to come to a decision about how to proceed. Before each scenario, the author introduces the scenario (e.g. what the scenario is trying to achieve functionally) to be solved by the participants.

The list of actions and communication techniques that may or may not be useful in completing the task are given before each scenario.

5.4 Evaluation Scenarios

1. *Grant CBPMS authority to community root*

Identify the Leader (the avatar is listed as owner of the land), and once identified let the Leader signify himself using a non-verbal communication technique that *would be understood to attract attention. Then walk onto a nearby structure that could represent an interactive object or area, to create the target tree*[8] *and grant the Leader authority to the root.*

Suggested actions and communication techniques:

- *Gesture '/yes' (nodding)*
- *Gesture '/hey' (wave hand up high)*
- *Walking*
- *Fly*

The four participants started on the parcel of land owned by the author where the 3D DMO was situated. Immediately three of the participants came together and began to use the chat function to solve the first part of the scenario. The following is an excerpt from their conversation:

> ...
> Participant #1: how do you find owner?
> P#2: it must be written on the land?
> P#1: where though?
> P#3: found it
> P#3: Right-Click on land and go About Land
> P#3: robin howlett's
> P#1: that's you
> P#3: has to be the hey gesture right?
> P#2: yeah
> P#1: yeh
> ...

[8] The target tree concept had been explained to participants before the scenario began.

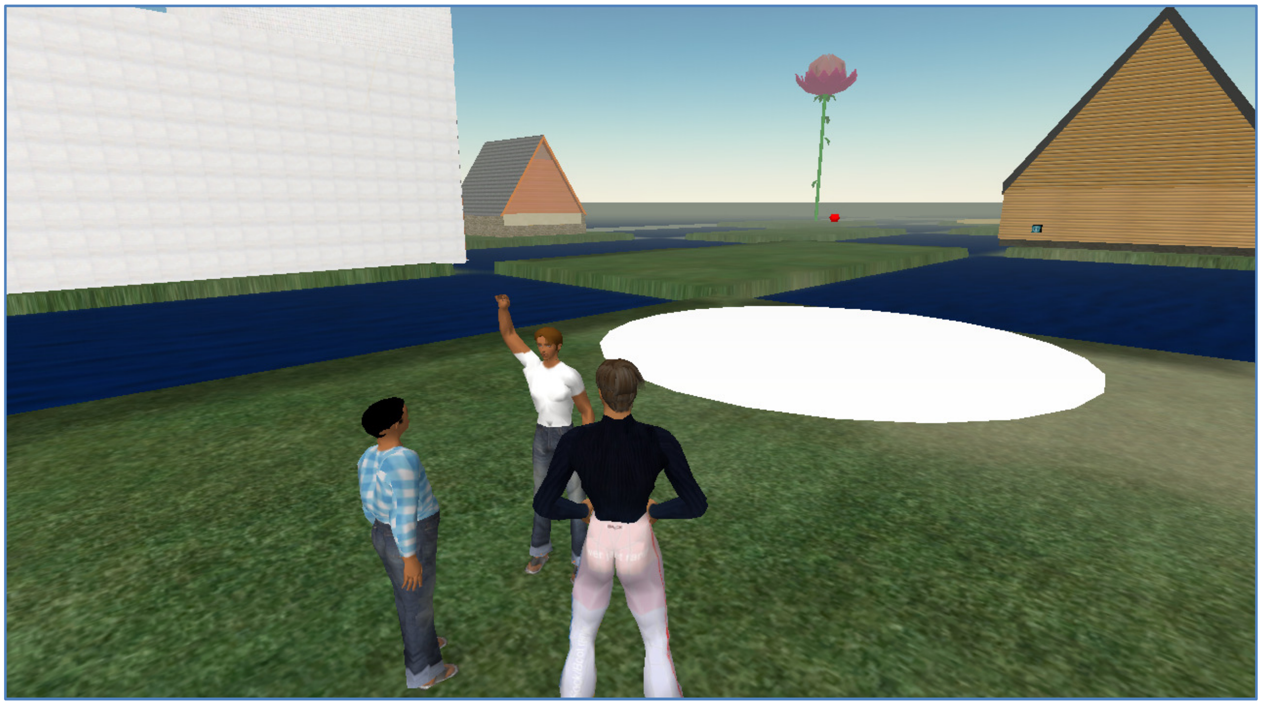
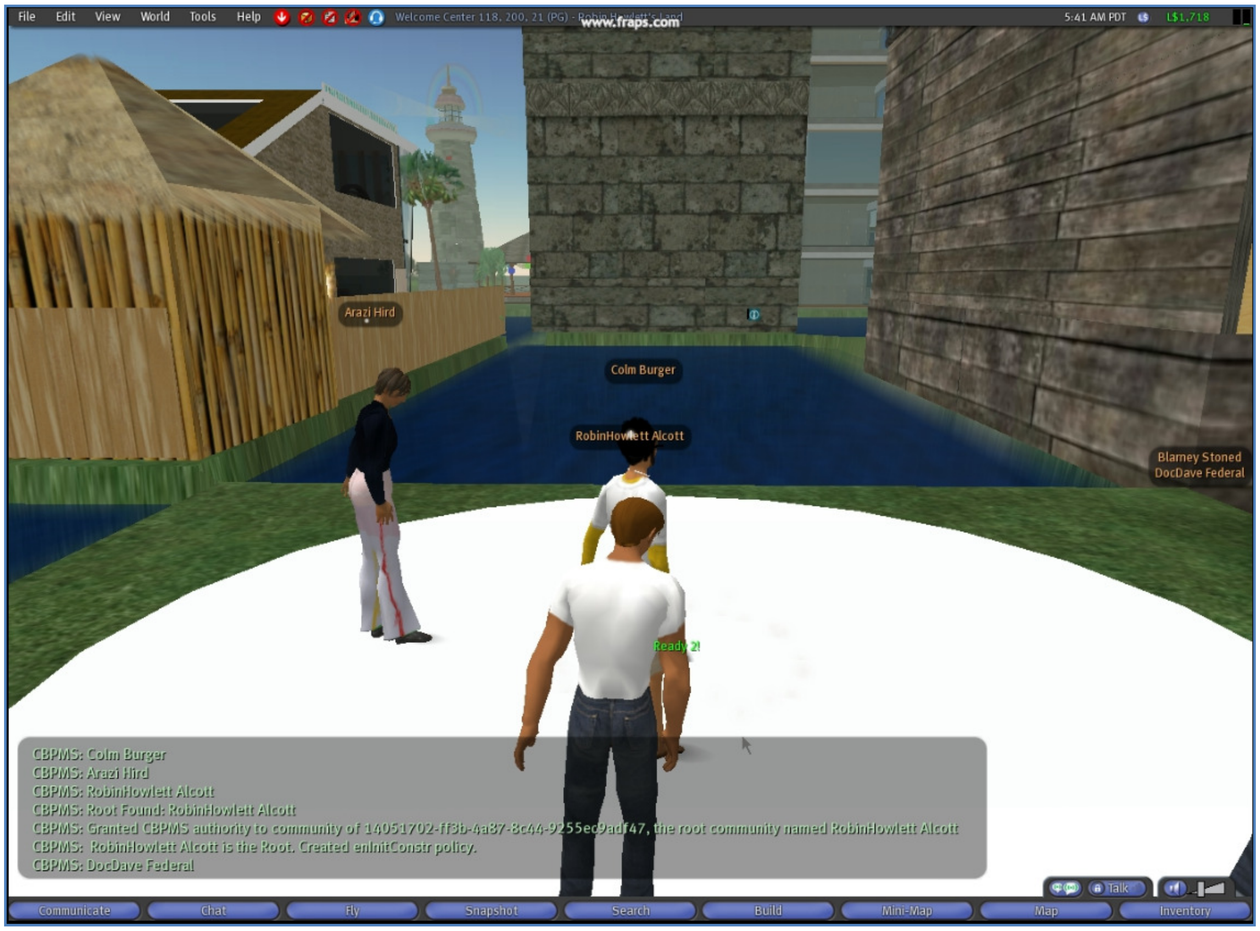


Figure 8: Avatar successfully identified

The group had then no difficulty identifying what the potential interactive structure could be and proceeded to walk over to the DMO (white circle).

The DMO had been initialised at this point in time. This meant that the resource model was in the process of being constructed. The Action Tree had already been built as explained in the *Implementation* chapter due to the fact that the nature of the conversion of the CBPMS functionality to LSL required hard-coding the action functions available. What had yet to be represented was the Target Tree and the DMO was waiting for the collection that represented the tree to be populated. This was solved by the community moving onto the DMO.

The DMO's sensors identified that avatars were in contact with it and it could identify which avatars they were, especially as one of them was its owner (by being the owner of the land the DMO resided on).



```
CBPMS: Colm Burger
CBPMS: Arazi Hird
CBPMS: RobinHowlett Alcott
CBPMS: Root Found: RobinHowlett Alcott
CBPMS: Granted CBPMS authority to community of 14051702-ff3b-4a87-8c44-9255ec9adf47, the root community named RobinHowlett Alcott
CBPMS:  RobinHowlett Alcott is the Root. Created enInitConstr policy.
CBPMS: DocDave Federal
```

Figure 9: Identifying the Leader and Captains

Since the owner is the Leader of the community, the bootstrapping default policy is associated his identification details and entered into the granted authority multidimensional collection. At this stage of the evaluation, the structure is currently a root (identified as Leader) with three child nodes (the Captains).

2. *Define global policies in community root*

Figure out who would be able to define a policy at this stage. Once identified, he can be referred to as "me", define a global policy called "forbidCBPMS" in the *root's community for the CBPMS resource (called "this") using the following information:*

- *The structure of the message is: policy [subject] [resource] [target_id] [action_id] [policyname]*
- *The top node of the either the Action Tree or the Target tree has an id of 1*
- *There are two children of the top node in the Action Tree; one relates the Community Actions (id=2), the other to the CBPMS resource actions (id=3)*

Suggested actions and communication techniques:

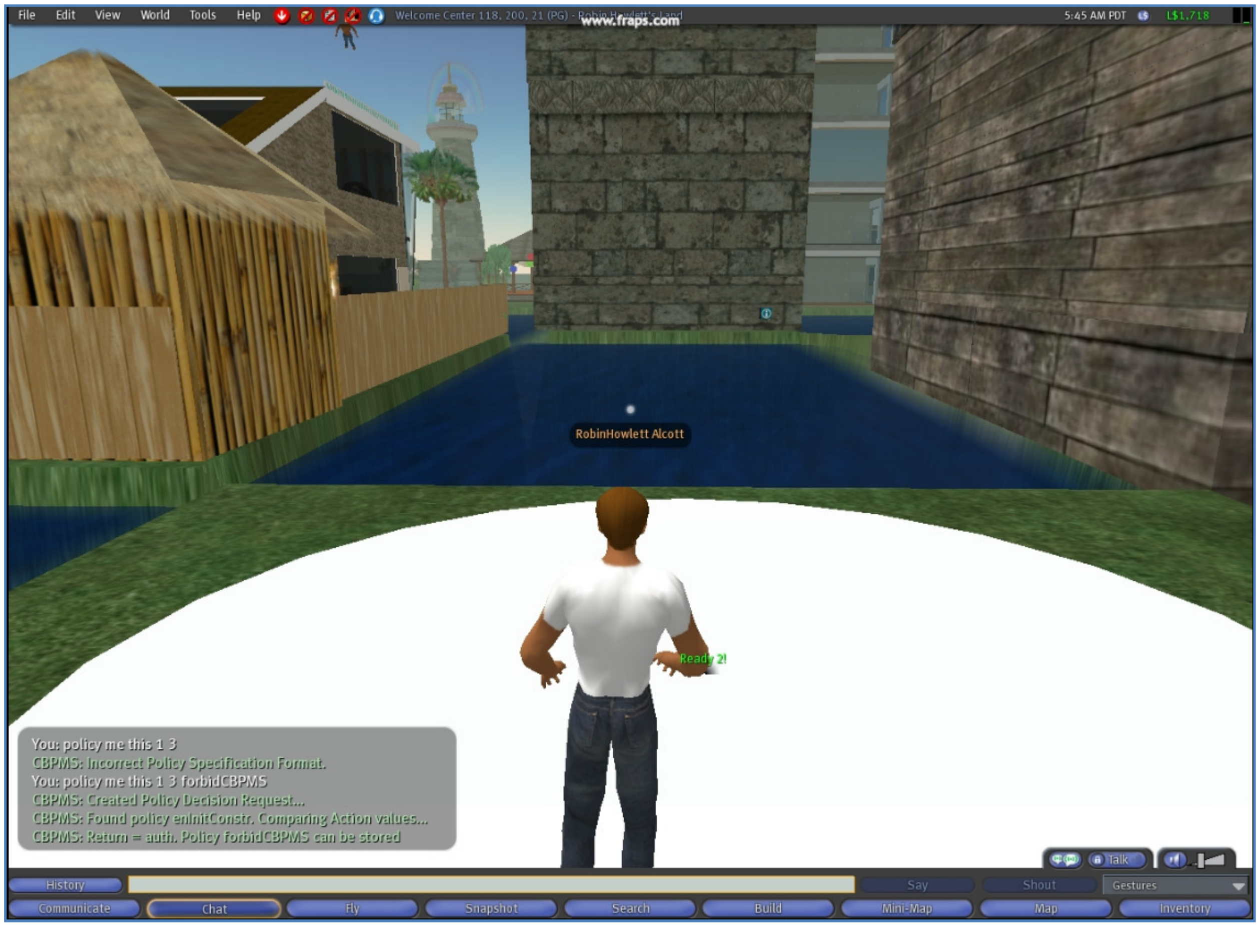
- *A Gesture*
- *Chat*
- *Transporting*
- *Walking*

The scenario began with the community's avatar placed off the DMO.

The community seemed to identify quickly that the root, avatar *RobinHowlett Alcott* would have been the only one capable of defining a policy as it has just been explained to them that only the Leader had been granted authority in the previous scenario. The group also felt that 'chat' was the only sensible way to communicate the policy message.

However there was much confusion over the idea of nodes on the resource authority trees being represented by numbers. After the reasons for storing the identifier's in this way was explained, the group eventually concurred that the target_id would be 1 (correct) and the action_id would be 3 (also correct). They also correctly believed the terms "me" and "this" to be recognised words within the policy reasoning service implemented within the DMO - the author confirmed that these details were correct to the community before the proceeded further.

Participant #3, who controlled the Leader avatar, walked on top of the DMO and typed out the policy message. However, the structure of the policy message entered was incorrect and an error message displayed such was broadcast as a text message to any object (including avatars) within a 10m radius. As the Leader was on the DMO, it received this message and correctly re-entered the message:



```
You: policy me this 1 3
CBPMS: Incorrect Policy Specification Format.
You: policy me this 1 3 forbidCBPMS
CBPMS: Created Policy Decision Request...
CBPMS: Found policy enInitConstr. Comparing Action values...
CBPMS: Return = auth. Policy forbidCBPMS can be stored
```

Figure 10: specifying a policy

3. *Delegate self-management authority*

The desired outcome for this evaluation is for a community structure outlined below to be configured.

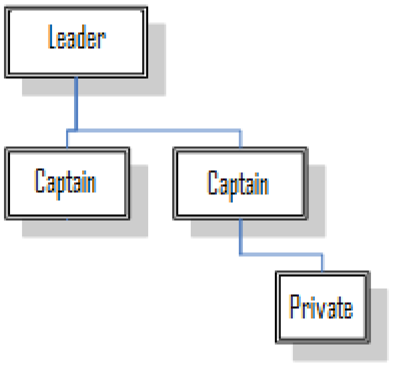


Figure 11: desired community structure

However, currently the structure is a Leader with three Captains. The avatar named DocDave Federal has been designated of class 'Private' and is therefore a child of the second Captain community node. By typing 'find' when within the boundaries of the DMO, the Leader can redefine the structure of the organisation based on the visual representation of the how the other members of the community are in relation to his own spatial position (a 'snapshot' of the avatars' positions after a ten second delay is used). Using only the avatars, have the community attempt to restructure itself to the desired organisation layout.

Note: no action or communication technique hints were given for this scenario as it was deemed unnecessary.

This scenario proved perhaps the most interesting as it seemed to have "clicked", as one participant put it, straightaway. The community first communication amongst themselves to organise whom was going to go where and then carried out the execution:

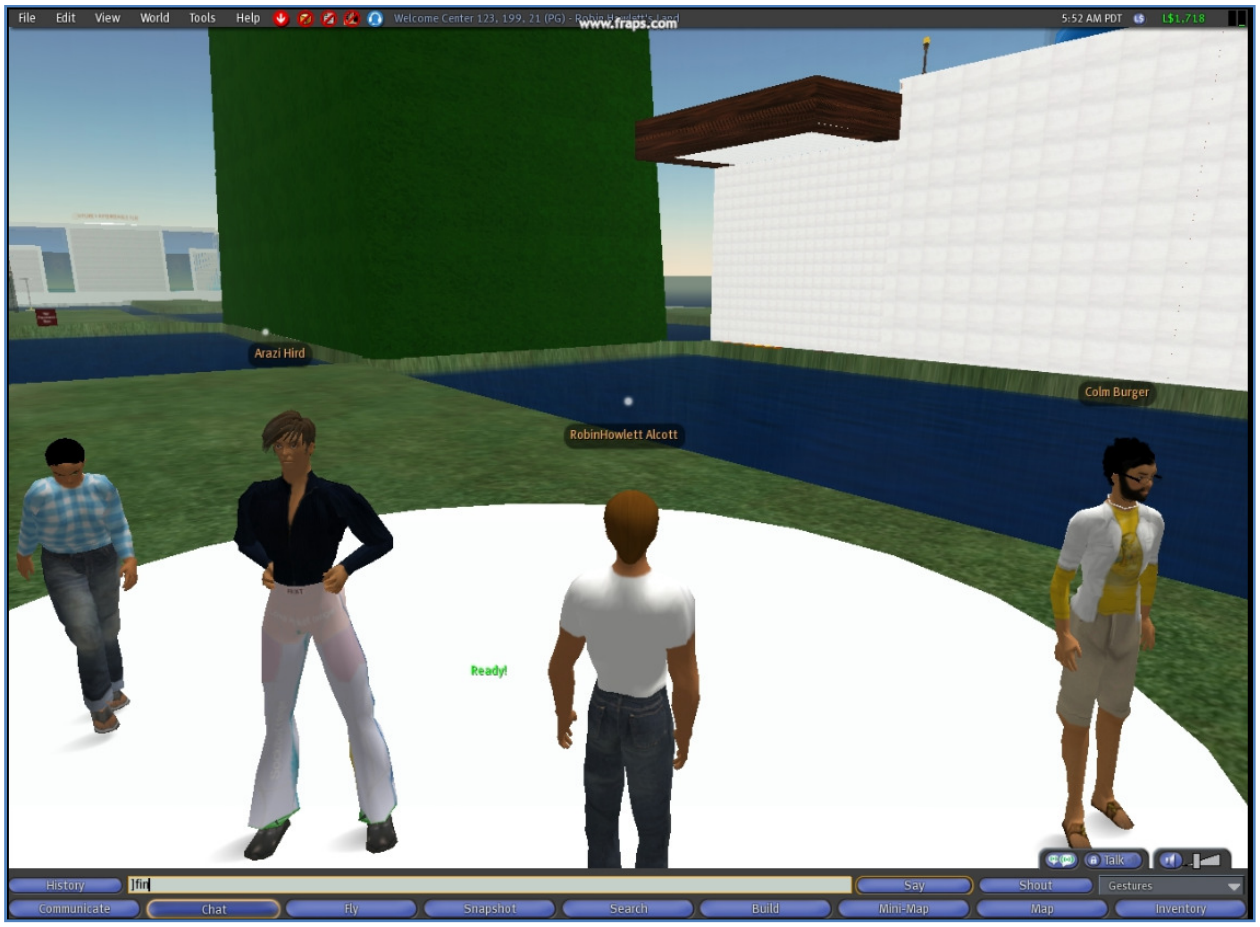
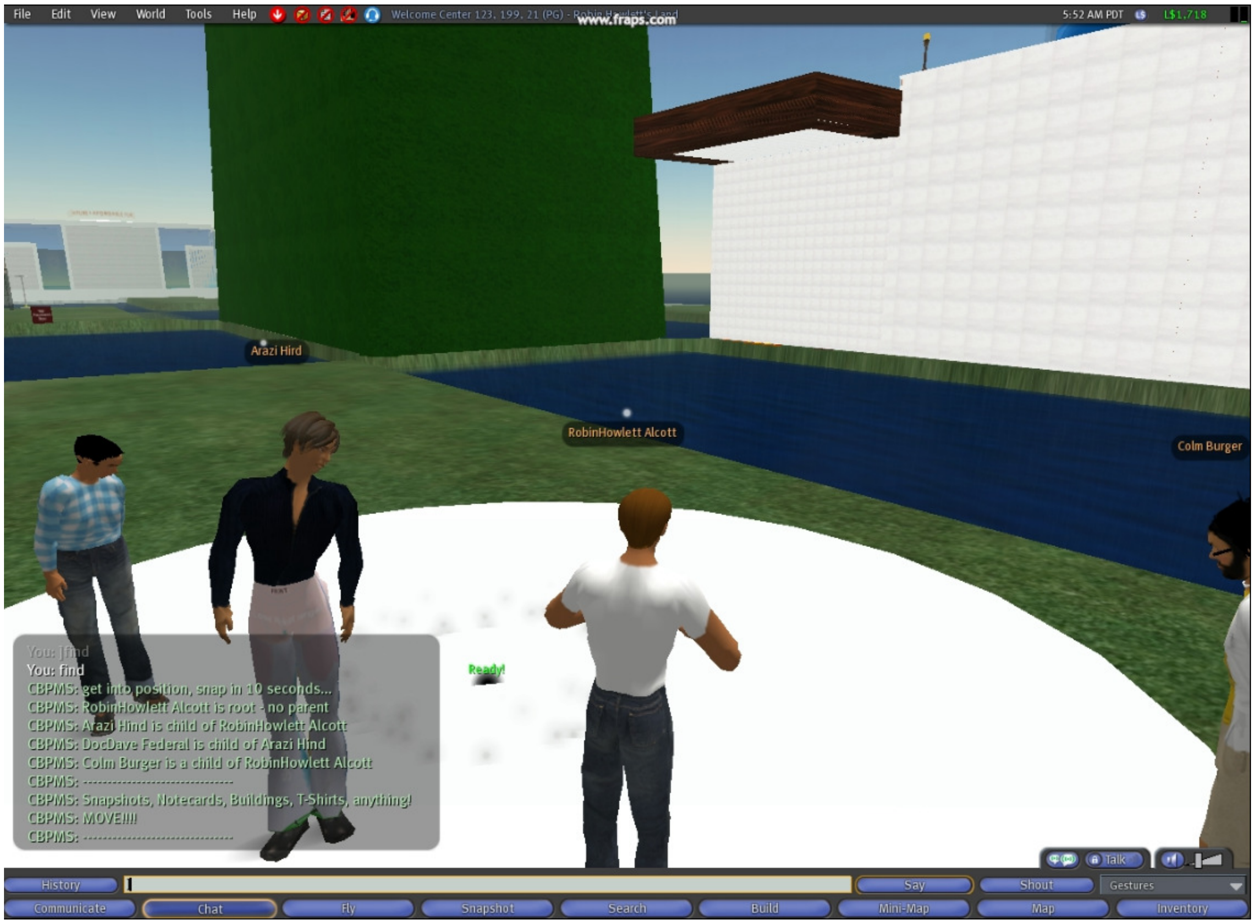


Figure 12: the Leader typing the 'find' command

The community found itself quite intuitive to organise themselves, although with smooth visualisation issues being experienced by some of the participants, getting the avatar to stop in the exact desired location proved frustrating to them.



```
You: ]find
You: find
CBPMS: get into position, snap in 10 seconds...
CBPMS: RobinHowlett Alcott is root - no parent
CBPMS: Arazi Hind is child of RobinHowlett Alcott
CBPMS: DocDave Federal is child of Arazi Hind
CBPMS: Colm Burger is a child of RobinHowlett Alcott
CBPMS: --------------------------------
CBPMS: Snapshots, Notecards, Buildings, T-Shirts, anything!
CBPMS: MOVE!!!!
CBPMS: --------------------------------
```

Figure 13: after a ten second delay, the snapshot is taken

After the ten-second delay, the DMO locates all avatars positions around the DMO's surface area relative to the Leader.

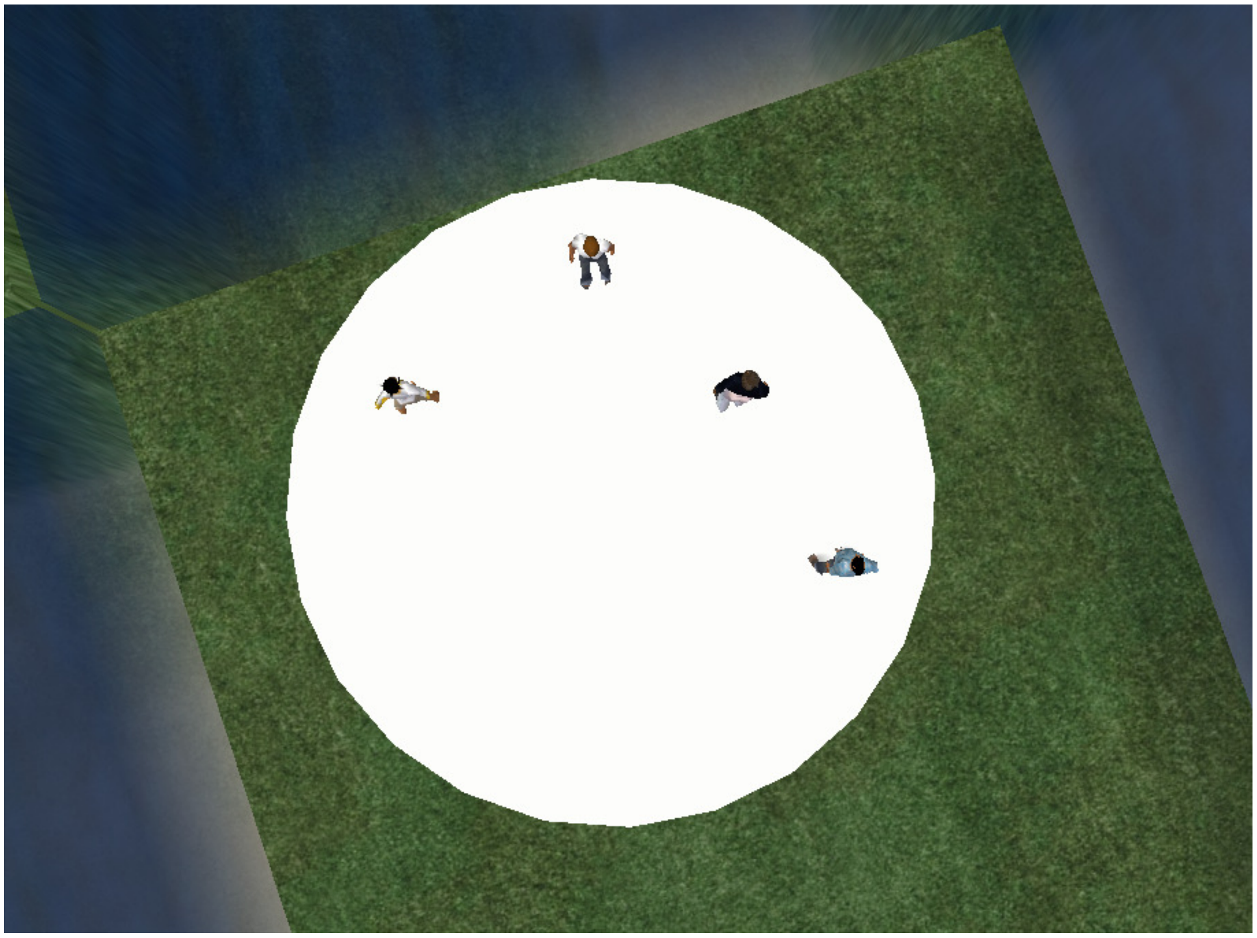


Figure 14: a top-down view, showing the desired structure through avatars

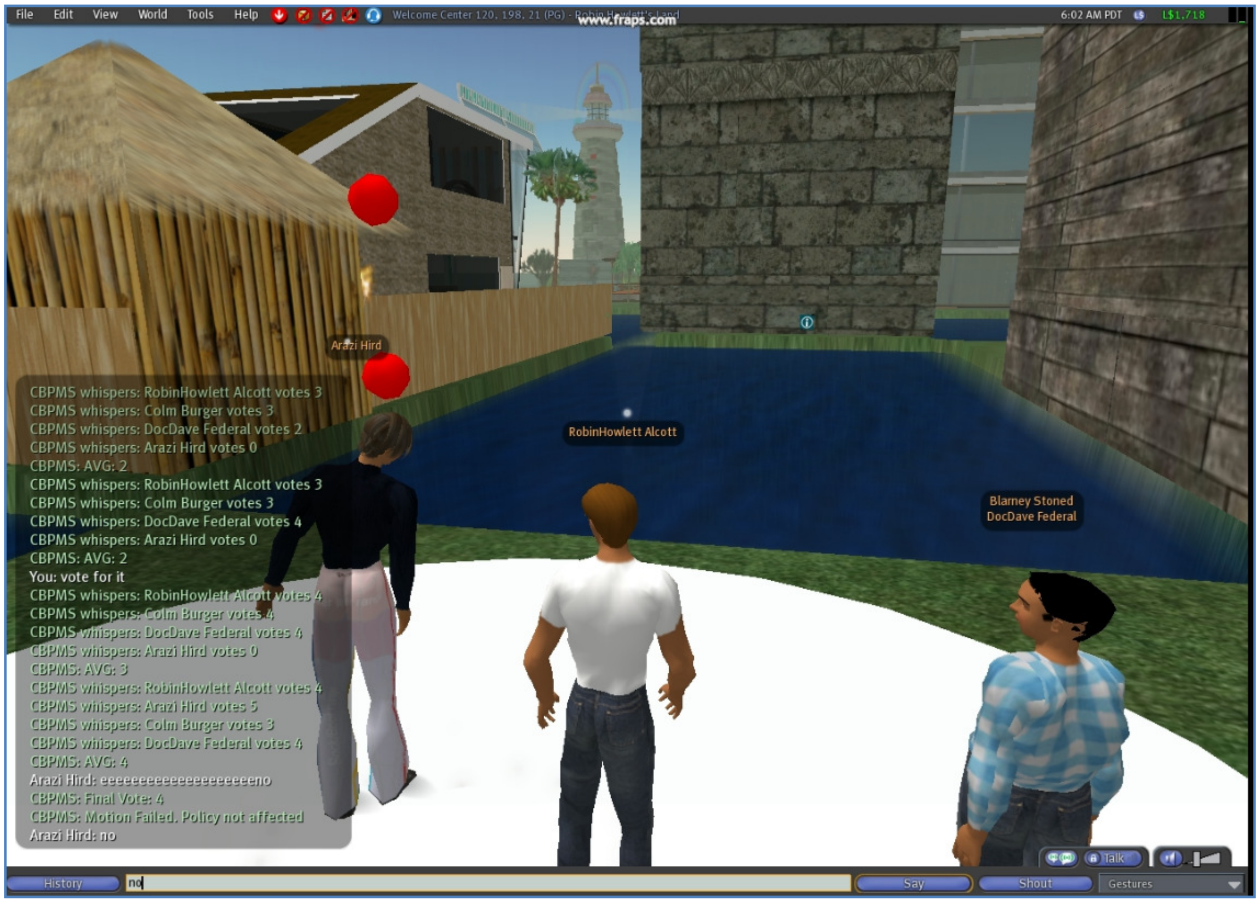
A spatial algorithm is used to use the relative positions to calculate which avatar is to be designated a child of another avatar. The *root* (Leader) will always have a parent value of the id of *All Communities*. By reconfiguring the structure like this, the policy management collection stored within the scripts will be updated with the new values and the traversed to ensure that they represent the new structure and authorisations have not been continuing to be granted to avatar based on a defunct structure.

4. Control Communities, Mandates & Autonomy

The objective of this scenario is to demonstrate how communities can integrate self-managed community policies with a collective decision-making procedure that needs to be approved before an action is authorized. The DMO also serves as a voting area where spatial information can be interpreted as voting intention and/or opinion. Difficult to articulate information like influence can be represented through physical manipulated of avatars and/or objects.

For this scenario, the community wasn't asked to solve a problem but just to participate in an active 15-second duration vote. They were requested to enter the vote at any stage they liked but before the end. There were also encouraged to attempt to discover if they could influence the outcome of a vote either directly or indirectly using any possible interaction.

The vote worked by calculating the positional percentile an avatar was stationed relative to the extreme North edge of the DMO. This was then converted to a value out of 10, where 10 indicated complete support for the motion, and 0 represented complete opposition. The DMO would keep the avatars informed of the current average value based on all participating voting avatar's positions. The vote that took place in this scenario was a simple majority-wins vote, whereby an average of more than 5 would constitute the motion under the vote passing.



```
CBPMS whispers: RobinHowlett Alcott votes 3
CBPMS whispers: Colm Burger votes 3
CBPMS whispers: DocDave Federal votes 2
CBPMS whispers: Arazi Hird votes 0
CBPMS: AVG: 2
CBPMS whispers: RobinHowlett Alcott votes 3
CBPMS whispers: Colm Burger votes 3
CBPMS whispers: DocDave Federal votes 4
CBPMS whispers: Arazi Hird votes 0
CBPMS: AVG: 2
You: vote for it
CBPMS whispers: RobinHowlett Alcott votes 4
CBPMS whispers: Colm Burger votes 4
CBPMS whispers: DocDave Federal votes 4
CBPMS whispers: Arazi Hird votes 0
CBPMS: AVG: 3
CBPMS whispers: RobinHowlett Alcott votes 4
CBPMS whispers: Arazi Hird votes 5
CBPMS whispers: Colm Burger votes 3
CBPMS whispers: DocDave Federal votes 4
CBPMS: AVG: 4
Arazi Hird: eeeeeeeeeeeeeeeeeeeeeno
CBPMS: Final Vote: 4
CBPMS: Motion Failed. Policy not affected
Arazi Hird: no
```

```
no
```

Figure 15: successful indirect influence of a vote that ended with a Motion Failed

An extra piece of functionality incorporated into the DMO is that when a vote is in progress, every time a chat message is broadcast, a ball signifying the colour of the current intent of the voter that spoke is *temporarily rezzed* (it will delete itself after 300 seconds) and slowly floats upwards. In the above figure, one of the participants has taken up a negative position to display opposition to the motion. However, to bring the vote average down, he needs to influence the other avatars to move to the opposition side. Hence, his chat messages were statements like "don't vote for it" and "no!" – the red colour of the balls signify his opposition to the motion for all to see and also identify him as an publically identifiable *active* (and vocal) opponent to the motion of the decision to be made.

One interesting note is that one participant recognised that these voting indicators were objects like any other and began flying after the balls after the vote finished to investigate what they contained (they words spoken, by whom and when) and began collecting them into his inventory.

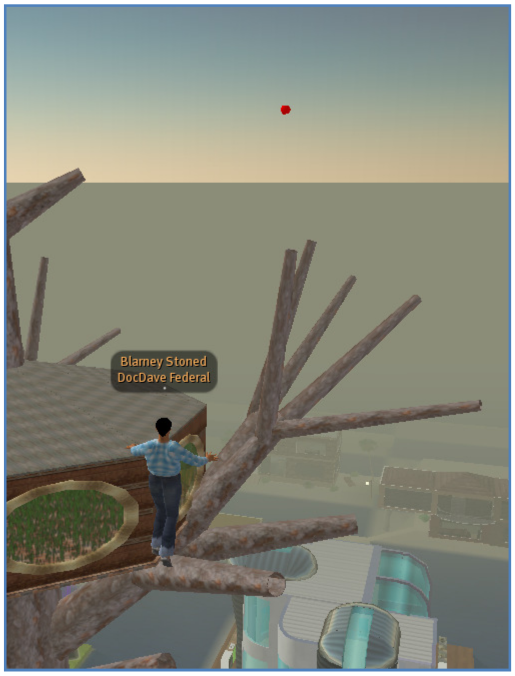


Figure 16: investigating voting indicators

During the evaluation, one participant relied on attempting to forcefully 'push' other avatars into a desired position – however the controls of the SL avatar through the SL Viewer were not fine enough for this to succeed and therefore the participant abandoned this strategy.

In the end, the chatting influencer succeeded and the vote failed to pass, resulting in no changes to the `canInvoke` policy authorisation for the Captains - that was the subject of the vote.

5.5 Discussion of Evaluation observations

After consulting the participants' feedback and extracting out the responses that corresponded to the five attributes of a reflective decision-making platform in a virtual environment for communities to self-manage, some interesting remarks were noted:

- **Spatial**

  All participants believed that the use of a three-dimensional virtual environment had significant benefits over a traditional form-based two-dimensional approach. Scale and space were consider easy to estimate and, although the majority of the scenarios were using abstractions, the success of the adoption of the 3D community structure indicated that increased dimensions meant increased levels of knowledge representation and decreased equivocality.

- **Cognitive**

  This turned out to be the most contentious point of the evaluation. Although participants recognised the levels of abstraction involved, there was a split over whether there were benefits associated with the cognitive nature of the community coming to a common decision or whether that cognition extended to the system as a whole. Also, one participant felt that the cognition being recognised by the other participants was merely a function of having a three-dimensional virtual reality to express one's "virtual intentions".

  The one common response felt was that without the author's explanations of the conversions made between the previous policy-based scheme implementation and the implementation of this dissertation, when that subject was approached by the scenarios, the participants would have been completely bewildered by the system as the available functionality is not displayed externally and is manipulated by actions such as avatar movement. Whether this is a due to the choice of implementing a sophisticated functionality like a Community Based Policy Management System or, perhaps more likely, a confirmation of the research presented by (Regian, 1997) that instructional strategies are often more influential that a change of media.

- **Immersion**
  The participants were questioned to whether they felt increasing comfort with the way the system worked and what it was trying to achieve. All participants felt that the environment contributed to vivid thoughts about actions that needed to be performed on abstract objects. All agreed that interacting with other human-like avatar seem unnecessary at times, however once the virtual environment became more abstract and conceptual, the increased knowledge representation available through a wider variety of communication techniques available to a distributed community within SL was highly beneficial.

- **Richness of Visuals**
  The participants recognised that due to the limited need for a large number of *prims* in the evaluation scenarios, the environment provided satisfaction visualisation of things like avatars, communities and structures, abstract representations like voting intention objects, colours and dimensions. All concurred that the Second Life Grid and Viewer provide a vast potential for advanced research into continuing work into graphical mental models and knowledge perspectives.

- **Interactivity**
  The participants' responses were all positive as regards to the range of interactivity options available to the user in Second Life. By blending

combinations of kinetics and a wide range communicative techniques to interact with a virtual environment, to come to a decision, and to manage things like dynamic community structural representation, the participants felt that the three-dimensional based system offered superior dynamism than compared to a browser-based form-oriented two-dimensional solution.

Overall, it can be argued that the participant’s responses conclude the following about the system:

- Increased knowledge representation.
- Benefit from the increased richness of perspective.
- Cognition is challenged by the abstract nature of the design.
- Equivocality is reduced.
- The interaction increases user internalisation.

5.6 Conclusion

These evaluation results have largely corresponded with the vision and research goals of the dissertation. A working system was constructed that could provide self-management functionality for a community who use cognitive decision making to collaborate within their 3D environment.

Chapter 6: Conclusions

This chapter describes ideas for future development of the system followed by overall conclusions about the design and the success of the project.

6.1 Further Work

This dissertation was an investigation into many new and highly complex areas of research, including providing abstract functionalities for knowledge representation, cognitive decision-making and community collaboration in a three-dimensional virtual environment.

While some the question posed by the vision and research objectives have been addressed, the nature of the project was very much like a proof-of-concept for a variety of different ideas and technologies.

In this section, further work is suggested for certain components of the system.

6.1.1 CBPMS

- Utilise the external full CBPMS implementation

There were far too many short comings with LSL to even consider attempting to extend further functionality of the in-world CBPMS implementation. The functionality of the policy-management system would be greatly extended if the external CBPM service was available in world.

6.1.2 Second Life Viewer

- Examine the SL Viewer’s source to identify if more conceptual modelling of communicative techniques can be modelled

The Second Life client application is officially open-source although there was a significant delay during this dissertation’s timeframe about whether the Viewer would be ported over to the Mono language.

6.1.3 Advanced Representations of Policies

One aspect of the potential of the system that the author is particularly keen on investigating is the use of external modelling tools to construct policies using non-standard spatial and construct representations as building blocks i.e. colour, shape, size, rotation etc

6.2 Conclusions

At the final stage of this research it can be said that this project has delivered a platform for reflective decision-making in a virtual environment.

Abbreviations

API	Application Programming Interface
CBPMS	Community Based Policy Management System
CDM	Cognitive Decision Making
CPDS	Community Policy Decision Service
CRMS	Community Record Management Service
DMO	Decision Management Object
IPK	Information, Preferences and Knowledge
LSL	Linden Scripting Language
MMOG	Massively Multiplayer Online Game
MUVE	Multi-Use Virtual Environment
PRS	Policy Reasoning Service
RL	Real Life
SL	Second Life
TOGA	Top-down Object-based Goal-oriented Approach
VR	Virtual Reality
XML-RPC	Extensible Mark-up Language – Remote Procedure Call

Bibliography

[1] Brdicka, B. (1999). *MUVE.* Retrieved from http://it.pedf.cuni.cz/~bobr/MUVE/muveen.htm

[2] Byron Reeves, T. M. (2007). *Leadership in Games and at Work: Implications for the Enterprise of Massively Multiplayer Online Role-playing Games.* IBM.

[3] Chen, Z. &. (2005). Enchaning UML Conceptual Modelling through the use of Virtual Reality. *38th Hawaii International Conference on System Sciences.*

[4] Craig R. Hall, R. J. (1996). *Virtual Reality for Training: Evaluating Knowledge Retention.*

[5] Harry, D. (n.d.). *Information Spaces*. Retrieved from MIT: http://web.media.mit.edu/~dharry/infospaces/

[6] Kemp. (2006). *Putting a Second Life "metaverse" skin on learning management systems.*

[7] Kevin Feeney, D. L. (2006). *Self-Managing Organisations.*

[8] L. Thompson, D. G. (2000). Avoiding Missed Opportunities in Managerial. *Organizational Behavior and Human Decision Processes* , 60-75.

[9] Larry McMaster, G. C. (2000). *Combining 2D and 3D Virtual Reality for Improved Learning.*

[10] Modjeska, D. K. (2000). *Hierarchical Data Visualization In Desktop Virtual Reality.* Toronto: University of Toronto.

[11] Quincy, J. (n.d.). *Voluntaristic Self-Determined Volitional Mandators Democratic Organization*. Retrieved from 1062.org: http://1062.org/

[12] R Sandhu, D. F. (2000). The NIST Model for Role-Based Access Control: Towards a Unified Standard. *Proceedings of 5th ACM Workshop on Role-Based Access Control, Berlin* , 47-61.

[13] Regian. (1997). *Virtual Reality for Training: Evaluating Transfer.*

[14] Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games. *Computer Magazine* , 25-32.