Awareness Mechanisms In Groupware Systems

Peter Byrne

A dissertation submitted to the University of Dublin,

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

2004

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.

Signed: ____________________

Peter Byrne

13 September 2004

PERMISSION TO LEND AND/OR COPY

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Saturnino Luz, for his valuable input and guidance through the project. I would also like to thank all my friends, family and classmates for their friendship, support and help during the year. Finally I would like to thank my brother for proof reading this work.

ABSTRACT

The main focus of this dissertation is to study the awareness mechanisms in groupware computing. The object of this study is to create a platform for testing awareness mechanisms in a general and empirical fashion. The platform will allow different awareness schemes to be enabled and disabled as required.

The awareness mechanisms that will be supported in this project are the use of colour as a carrier of embodiment information, the use of radars and telepointers to present location awareness information in the workspace, the use of landmarks as a shorthand for navigation and reference in a shared workspace, and finally the effect of different levels of communications on the tasks.

This study will conduct a pilot trial with groups of 2 to 4 users to test the effectiveness of the platform developed. Preliminary results from these trials will be summarised.

Table of Contents

TABLE OF FIGURES

1 INTRODUCTION

The main focus of this dissertation is to study the awareness mechanisms in groupware computing. It is hoped, in the course of this study to find some empirical results that will show that awareness mechanisms significantly affect group interactions and that they should improve the effectiveness of group working. In previous studies awareness mechanisms have been studied in the real world by observing how people in the same room can work together to solve various problems [5]. In this pilot study much of the same ground will be revisited, the users will be isolated from each other so that their only communication will be through computer communications. It is then to be hoped that by incorporating analogue functions that will approximate real, or enhance traditional awareness mechanisms, we can observe improvements in productivity and accuracy of the team.

In earlier work the main focus of the research was on text based collaboration. This study will move away from this to a certain extent and try to abstract away some of the key artefacts that are present in a textual document creation and then try to create a more generalised framework in which collaborative writing is but one part. This study will present a game in which the user groups will have to collaborate to solve a simple maze problem. It is hoped, by conducting a number of experiments in this way using small groups of 1-4 users, to measure their performance as various awareness features are added or removed, then observing what features are most used and in what respect they proved useful. User feedback in the form of a survey will be collected to gather their thoughts and experiences on the various mechanisms used. If this proves useful this could then prove a solid basis for a future larger study into awareness mechanisms using this platform.

1.1 Why Not Text?

There are several problems when trying to do awareness studies when the medium used is text. There are a number of problems inherent in the content of a text document that is being composed, in [24] it states "participants in joint projects bring with them expectations about joint writing". For example, suppose two colleagues are working together to write a research paper and one of the authors concentrated mainly on the practical research and the other author concentrated on the evaluation of the findings.

In this scenario it might be possible for the two authors to write the document with little consultation with each other as the first can write the general study and the methods used to arrive at the results and the second author can write the findings and conclusions to the paper. They may only have to consult to harmonise a few sections. Note in this particular scenario there would not be much collaboration and it would not be a very helpful scenario as the authors are mostly working on completely different sections of the document.

When studying awareness schemes, it would also not be a very helpful scenario as the authors would need very little communication at the time of writing the document. In this case most of the awareness schemes outlined would be fairly irrelevant. Similar scenarios can be found in different textual domains and therefore it is an almost intractable task to account for all of these special cases. It would be better if we could abstract away, to some extent, the underlying medium and concentrate on the awareness mechanisms and interaction between the users.

It is by this reasoning that it was decided to try to investigate another situation where the awareness mechanisms could be studied in a more abstract and hopefully more empirical fashion. Obviously there is still a case for studying awareness mechanisms in the textual domain or for that matter any particular media domain in collaborative working, especially as the particular algorithm might be optimised to this particular domain. By observing particular features that might be more appropriate in each case. Here we believe that by studying these awareness tools in a more abstract manner you would gain an insight into the more general features of collaborative working, which could be applied to particular domains as appropriate.

As the tool chosen is a game system that none of the users will have any real a priori knowledge, they will not be going into the task with any inbuilt assumptions about how to collaborate, this would not be the case if it was a writing task as most of the writers will have acquired a writing style and a paradigm for collaboration that already works for them and may be unwilling to change. In this new situation the users will be provided with greater freedom and more scope for collaboration.

1.2 Background and Context

The area of groupware or computer supported collaborative working is a very large field of study. In [8] they broke the area into several parts including, Message Systems, multi-user editors, group decision support systems, electronic meeting rooms, computer conferencing, intelligent agents and coordination systems. They also identified two boarder categories in which these groupware systems can live and operate, real-time synchronous versus asynchronous systems. In [16] we again have the distinction between synchronous and asynchronous, they also break the area into four key distinct areas, message systems, conferencing systems, meeting rooms and co-authoring and argumentation. It is the area of multi-user editors, co-authoring and argumentation systems that the research aims to look at the most closely.

The area of groupware can be attacked from a number of different perspectives, from [8]; we get the distributed systems perspective the communications perspective, the artificial intelligence perspective, and the social theory perspective.

When trying to tackle a problem in the Computer Supported Collaborative Working, CSCW, field, it is important to bear this spectrum of views in mind as there is a strong interplay between them all to one degree or another [11] with emphasis on the users of the system and how to support natural interaction. Systems that allow users to work together across time and space divisions have been envisioned and a classic classification of the systems can be seen in the table in [8].

Messaging Systems grew out of the domain of email like systems, and can be thought to consist of systems ranging from asynchronous system like email, voicemail support systems, to instant messaging and even SMS text-messaging systems. This is a useful paradigm and has been included to varying degrees into almost all groupware systems. The ability to contact your colleagues efficiently is needed in any group of people, regardless of time. It is also very helpful to be able to track messages in the system and retain old messages for context. Example can range from threaded discussion groups to the ability to archive old conversations in instant messenger systems.

Conferencing systems include systems ranging from simple real-time tools like MS net meeting to more sophisticated systems involving integrated phone systems and webcams, use of shared workspaces and whiteboards, where "the goal is to improve the productivity of decision-making meetings, either by speeding up the decision-making process or by improving the quality of the resulting decisions" [11]. Other tools in this area are voting tools or tools for issue analysis and idea generation like brainstorming tools. These system can allow users "who are either gathered in an electronic meeting room or physically dispersed, to interact synchronously through their workstations or terminals" [7] or through an audio link.

Meeting rooms systems are related to conferencing systems in this way, these systems can be larger scale systems that may contain a physical component for example, the PlexCenter Planning and decision support laboratory at the University of Arizona [17], but can be purely virtual environments.

Collaborative editors systems are tools to support the creation of documents, text, images, or other shared object, by a group of users. These systems can again be synchronous or asynchronous environments [7]. In [24] they have provided a survey on the writing strategies used by different writing teams and conclude that the collaborative editor should try and accommodate as many of these differing styles in order to offer a working environment that feels natural. This study [24] also shows the value of understanding the users needs before designing a collaborative writing system.

In this dissertation I will be concentrating on evaluating concepts from the collaborative editor field. In order to undertake a successful collaborative writing project, various support infrastructures should be provided [18]; this would take the form of concurrency control systems to reduce conflicts as users interacted with shared artefacts, documents, images, etc. Another element of infrastructure is consistency maintenance systems, several schemes have been proposed. Currently one of the most studied types of schemes is Operational Transformation schemes. These operations allow loose consistency constraints, i.e. they can operate without the need for strict locking [1], [2], [3] often these schemes try to resolved conflicts automatically through intention preservation systems.

These systems have now been extended to include support for exploiting the structure of documents to help with reducing conflicts in the system. For example in [4] they have created an Operational Transformation algorithm that can be used with XML documents. By using tree based documents a lot more concurrency can be supported. In a sequential flat file system the user's actions will have to be ordered and transformed against all actions as a change to any part of the document can affect the rest of the document.

Avoiding conflicts in a shared workspace in collaborative editing is a difficult problem to solve using algorithms alone, no matter how complicated, or strict, for example locking [6]. In [11] it asks if it would not be more efficient to let social protocols reduce the amount of conflict and the need for locking etc. under the principle of minimising constraints they identified.

Another approach hoped to increase collaboration in CSCW systems is awareness mechanisms. Awareness has been defined as an "understanding of the activities of others, which provide a context for your own activity" [14]. This has been mostly studied in terms of collaborative editing but it also has been studied in other contexts, for example in [15] awareness in air traffic management, aircraft carrier operations and space shuttle mission control. There are several different aspects and levels to awareness.

In [5] a study was made into the awareness mechanisms utilised by several small groups which had to complete small collaborative tasks involving simple games in a shared physical workspace. In this study several different aspects to awareness are identified, including Situation Awareness and Workspace Awareness.

Awareness can be summed up as knowledge a user has about a system and the users in that system that help them with the task in hand.

With the expansion of information technology to almost everywhere, with its enabling technologies of high specification workstations and reasonably reliable networking, the opportunities for increased collaboration on projects is growing. In this context users require a shared domain in which to operate collaboratively. Typically this is called a workspace as opposed to workplace as it is a virtual environment in nature and can be accessed from almost anywhere.

“Workspace denotes the system designed to support collaborative work, rather than the physical location or workplace where that system is used” [19]. Using collaborative editing environments you have a less rich set of communications mechanisms available than for traditional face-to-face communication. A lot of the awareness information that you get for free in a face-to-face interaction is lost when the communication is mediated through a computer system.

Groupware interfaces can often be awkward and difficult to use [20], particularly the first time. Often the collaboration element to the system has in effect been bolted on to an existing single-user application, and this can lead to badly designed interfaces. Designers are seeking to recreate the rich experiences of face-to-face communication in Groupware applications, and this involves giving the user greater flexibility in their working styles and interaction methods. This is where this study comes in; it should prove useful as a testing environment for evaluating various different groupware paradigms in an empirical and systematic manner that will eventually help designers create better groupware applications that are easier and more user-friendly to operate.

1.3 SCOPE AND OBJECTIVES

The main objective of the project is to build a system that can be used for testing various different awareness mechanisms and tools. This system will take the form of a simple multiplayer game in which various awareness systems can be turned on and off. The particular awareness schemes that we wish to test eventually are document radar systems, out of band communications (both textual and audio), contextual identification systems, telepointers, and colour as a carrier of embodiment information.

Eventually, these systems will be tested in various different combinations to see what the presence or absence of a particular awareness and communications mechanisms does for the interaction and performance of the user groups. The system will also be evaluated in a pilot study to try and gather some preliminary results in the form of general user behaviour and how they interact with some of the awareness schemes.

For example what conveys more awareness information radar systems or telepointers etc? It is hoped the system will be able to be extended to gather further empirical results in the form of some usage statistics, for example, how long users are observing the same section, or how often the telepointer features are used when available.

It should also be possible to draw an analogue with different writing styles by using the system. It could be simulated by imposing protocols on the users or restricting the available actions of certain users. For example, if we only allow one user to see the whole picture using the Map screen, while all the other contributors will only see small sections of the board at any one time. It will be up to the person with the whole view to coordinate the contributions of the team. This would be analogous to the scribe style of writing [24].

The testing framework developed can be use in future for detailed empirical studies of the affects and usage of a spread of awareness mechanisms and communications mediums in computer supported collaborative working.

1.4 Overview of Dissertation

The main work in this dissertation involved creating the framework tool for testing the awareness mechanisms we feel would be interesting to study in detail.

The tool developed was a platform for testing. It consisted of a simple multiplayer game that could be used over a network. The game was written in java and operated over RMI. The tool would support several different awareness mechanisms and would be able to operate most effectively in a LAN environment but could be used over the internet but the latency would be a performance bottleneck.

The tool has a number of key features that would be of interest to future researchers. The first element that merits a mention is that it is easy to distribute, it is a single executable jar for the clients, and a one executable jar for the server. Another feature is the simple map editor tool that will allow users to create map scenarios that they think will foster the desired communication and interaction that they are interested in studying.

The game itself can be configured to allow particular configurations of awareness mechanisms. Note that as these are implemented at the interface level it is possible to manage these awareness mechanisms on a client by client basis. Also the views of the system can be enabled or disabled on a client by client basis.

The document will take the following format, beginning in section two with a summary of the current state of the art and an introduction to the field. Then section three will comprise a description of the implementation, outlining the key design decisions building the awareness platform. Section four is an evaluation of the awareness platform with preliminary results from the pilot study. Finally the dissertation closes with the conclusions and future work.

2 STATE OF THE ART

Awareness, as a concept in general, is related to the individual's perception of the workspace which they can use in order to complete a task. The greater the awareness of the workspace the more able the user should be in completing the task efficiently.

In [20] they conducted a small scale experiment comparing the effect of using a multi-user collaboration environment to perform some simple tasks. They had the users conduct these tasks using two different modes of the application. The first time without awareness support and then later with the awareness support in the form of radars and telepointers that allowed the user see what the other users in the workspace were doing. They concluded that the times to complete the tasks were lower when they used the awareness schemes. Noting that "the difficulty is particularly acute when the workspace is larger than the screen and people navigate independently through the workspace (called relaxed-WYSIWIS view sharing"[20].

The testing platform developed for this project aims to allow testing of a similar set of awareness resources but should enable researches to evaluate more awareness tools in an empirical fashion.

"While staying aware of others is something that we take for granted in the everyday world, maintaining this awareness has proven to be difficult in real-time distributed systems where information resources are poor and interaction mechanisms are foreign. As a result, working together through a groupware system often seems inefficient and clumsy compared to face-to-face work.” [5]. Here it shows that the interaction through a computer interface can be a less rich experience that face-to-face communication as most of the awareness information that you get for free in face-to-face communication is lost. Groupware tools in order to be more natural to work with should try and restore this lost awareness information to the users. It also calls for these features to be considered from the beginning, not just added to existing tools where possible.

It is important to note what kind of awareness information is collected by people and how to present awareness information in a suitable interface. In one study [5] they identified three key ways in which awareness information can be garnered.

The first method is Consequential Communication of awareness information, this is information that is conveyed by a user’s actions in a systems. To take the example from the paper if a pilot is flying a plane and reaches over to engage the landing gear, the co-pilot will see this and will be informed that the landing gear has been engaged without any explicit communication necessary, he only has to observe the pilots actions, or interactions in the shared workspace.

The second way of gathering awareness information is through the mechanism of feed through, this is information gathered from the system as the users interact with the objects in the shared context. For example if you cannot see the user performing the actions directly you can perceive that they have done something to the objects in the system purely by observing the changes in the shared objects, caused by the other users actions. As an example from [5] in an air traffic control system, the person controlling the Departures may not be able to communicate or see the person who is controlling the arrivals of the planes. By observing the display listing the arrivals will be able to ascertain that the arrivals controller is landing planes as the entries on the Arrivals board change to Landed.

This can also be seen in collaborative editing environments as writers contribute to the document, the other writers can tell what the others doing by observing the document getting longer and new text appearing etc.

The third way listed is through intentional communication. This occurs when explicit communication is used. For example by sending a message or calling to another user in the system. This would be traditional communication and is probably the easiest to capture in a computer environment. There are several different options for this explicit communication, including email, fax, instant message, phone call, electronic conferencing tools and so on.

What kind of information can be captured in awareness enabled systems? The information we would like to show, "at a simple level … involves knowledge of who is present, where they are working, and what they are doing" [20].

From [5] we get a more detailed set of information that is required in the present for workspace awareness and contextual information. Under the heading of "who is in the system?" three distinct categories emerged, firstly is there anybody else using the workspace, presence. Following this, what is their identity, and thirdly who did what.

Under the heading of "what are the users doing?" they considered what actions are being performed, what are the other users in the system's intentions, and what document or more generally artefact or object are the other users interacting with.

Finally under the heading of "where are the users in the workspace?" should be considered the other users current locations in the workspace, what they are currently looking at and where they can potentially view, and what they can potentially interact with. For example in some workspaces there may be roles associated with the objects in the systems, so that only certain users can see or use these particular objects.

2.1 IDENTITY AND AWARENESS

Awareness information, in order to avoid information overload and feel more natural, should be located at the periphery. The periphery is "what we are attuned to without attending to explicitly" [21]. "Ambient awareness displays use the tactic of embedding information into the user's surrounding environment, often without using standard computer screens, and often utilizing the senses of sound and touch (in addition to vision)."[14]

One of the first awareness techniques that we wished to evaluate was the use of colour to represent users in the systems. It was felt that this would be a non-obtrusive and intuitive system. "Users should have an embodiment; many existing interfaces fail to adequately embody users within the common display space" [22].

Embodiment can be any way of consistently representing a user in the workspace and "describes the way in which users are themselves directly represented within the display space" [22]. In this project the method of embodiment is to associate a different colour with each user and then apply this consistently through the interface. This has the advantage that after a while the users can easily see who is doing what. This information is in the periphery. "Identity within a groupware system is another kind of awareness. ... Identity makes the processes of workspace awareness and feed through possible by supplying actors for the actions." [23].

Embodiment could have been included in the system using additional features but at this point was passed over. One method as outlined in [29] was to use pictures of the contributors in the radar view of the system, or possibly video. It could also be extended to recording the history of the system, as an example the users could move the mouse over an object and a picture would appear in a tool tip like manner displaying the user who last interacted the object.

2.2 Radar and Telepointers

Two of the most popular awareness mechanisms are radar and telepointers. Radar views involve showing a representation of the view-ports of each user in a common map of the total workspace. With a radar view it is possible to ascertain where another user is currently looking. This addresses one of the key awareness questions raised earlier. For tasks that require information and activity, especially where it is difficult to describe the workspace verbally, the radar view reduces completion time and increase user satisfaction [20].

In What-You-See-Is-What-I-See Systems all users have the same view of the shared workspace. But WYSIWIS systems can be overly restrictive and do not readily support users who wish to work on individual tasks at times, but it does have the benefit that all users in the system have the same view of the system so have the same shared context and automatic appreciation of what the other users are looking at.

With relaxed-WYSIWIS groupware you lose this automatic awareness of the shared context. With such systems the users have better individual control so there is a trade off between better support for individual work and the amount of awareness [29]. To try and improve this, solutions involving overlaying the individual views of the workspace over the entire workspace help to restore this gap in awareness and should increase group awareness.

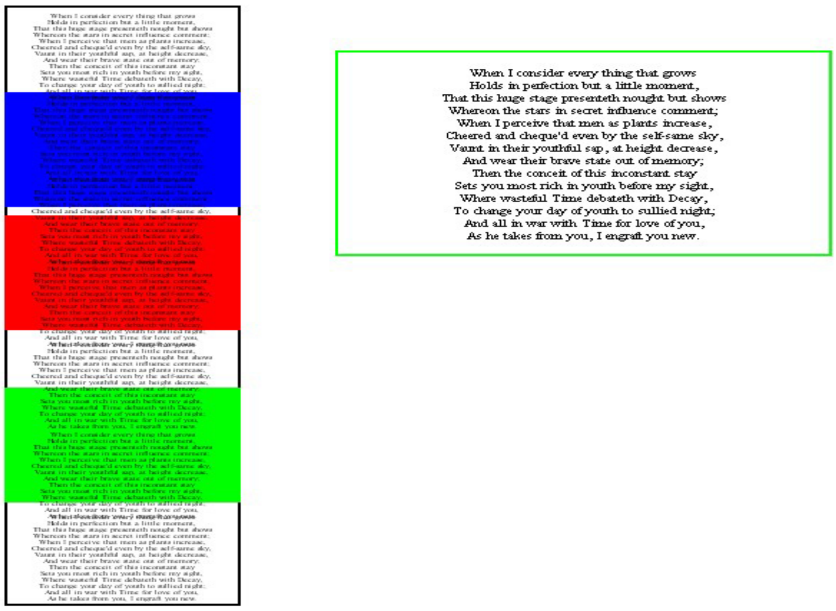

When I consider every thing that grows
Holds in perfection but a little moment,
That this huge stage presenteth nought but shows
Whereon the stars in secret influence comment;
When I perceive that men as plants increase,
Cheered and cheque'd even by the self-same sky,
Vaunt in their youthful sap, at height decrease,
And wear their brave state out of memory;
Then the conceit of this inconstant stay
Sets you most rich in youth before my sight,
Where wasteful Time debateth with Decay,
To change your day of youth to sullied night;
And all in war with Time for love of you,
As he takes from you, I engraft you new.

FIGURE 1: DOCUMENT RADAR

One thing that might be interesting to study is that when working in a collaborative environment, what percentage of the time users spend looking at the same view. This would be interesting because if it was a high proportion of the time the use of relaxed-WYSIWIS might not justify the loss of group awareness.

“Telepointers are one of the most useful elements of real-time groupware. They are simple and computationally inexpensive, but provide embodiment, awareness, and gestural communication.”[30]. Telepointers can be cursors that track the location and movement of other users mouse pointer, or they can be used in a higher granularity mode where they operate as a remote pointing device. They can provide important information in real-time groupware applications. Telepointers can be used to provide embodiment, granting awareness of other users in the system. Telepointers are cheaper carriers of embodiment information that video or image representations in the workspace [30].

The main problem with telepointers is that when used over non-ideal network setups that can be slow to react and will end up displaying out of date information. In [30] they describe a High Performance Telepointer implementation that can get around this problem. This system includes some advanced features like motion prediction and measures for improving pointer accuracy.

2.3 SHARED CONTEXT

Participants have to know the environment or workspace they are operating in to some extent. The users should be able to remember or recognize certain spatial features of the workspace.

Having Information like size or bounds of the workspace and what the current state of the users and objects in the system is considered very important [33]. Some virtual environment systems even have learning about the workspace as one of the design goals. It is useful if the users have some landmarks in the system for discussing the system, these landmarks or common contextual reference points to aid communication as they are a shortcut to locations in the document and can be expressed orally rather than rely on telepointers which might distract from the current task you are performing [33]. In this study the use of a coordinate system is there to provide the users with this context that they can use to inform their discussions about the workspace.

“Where users are involved in a joint project of some kind, such as putting together a machine, viewing a shared document, or hunting down a monster, people need not only to see the environment and each other, but also to communicate in order to convey their own intentions, confirm the understanding of each others’ intentions, and coordinate their joint actions.”[34].

So it is useful to have reference points in a shared workspace for communication and navigation. In an editable document the reference points could be headings or other standout features of the document. In a collaborative programming environment the system used could be line numbers and class names. In a collaborative drawing environment the reference points could be coordinates or standout shapes or objects in the drawing.

2.4 COMMUNICATION

Communication is obviously very important in groupware applications as collaborative working is all about keeping everybody informed with as much information as possible. Using a computer system gives you access to a rich set of possible communications mechanisms. It is deciding which solution works best for a given situation that is the problem.

The first method of communication under consideration is the use of text based communications; this typically takes the form of an instant messaging system. During the study of computer supported collaborative writing it was noted on several occasions that it would be very useful if an application could support some form of informal communication. This was said to be useful for clarifying design issues and helping with conflict avoidance. The benefits of such system are also described in [25]. Therefore it was decided that it would be a very good idea to look further into this issue of out of channel communication.

People from different area often have different interpretations of concepts; communication systems can help negotiate a common understanding. In [25] it states the following "CSCW systems should have some secondary mechanism or communication back-channel to allow users to negotiate the norms of use, exceptions and breakdowns among themselves, making the system more flexible".

During the course of normal working operations, exceptions are found to be routine. It has been observed that much of office work is handling exceptional circumstances. Augmenting CSCW systems with chat and other communication systems help to deal with exceptions and improve fluidity.

The above applies as much to verbal communication as to text based chatting. We hope to conduct a number of studies into various combinations of face-to-face communication, verbal communication and communication through the medium of text. "Results also reveal that the computer-mediated teams significantly outperformed face-to-face teams in the idea-generation task." [31]. One of the reasons for this improvement is that the computer supported system retains history of what was said [31].

In one study [26] they recommend that chat or instant messaging systems should be in context of the objects being discussed. They advise that the chat client is built directly into the groupware application and not in a separate window as has traditionally been the case. In another study [32] they found that users communicated more and in a more natural way when dealing on coactive tasks rather than tasks where they could easily partition the workload, and were only encouraged to communicate.

3 IMPLEMENTATION

In this section I will outline the main problems involved in implementing this project. I will explain the key design decisions made and hopefully one should be clearer on what the project actually involved. This section should also explain decisions like what technologies where chosen and why and finally, I should give a detailed description of particularly important components of the system designed.

The system built is a simple multiplayer game that can be used to study certain facets of Human Computer Interaction and interpersonal communication and interaction. As explained earlier, the system would record user's actions and communications and it is hoped that this information can be used in, and taken into consideration when, designing groupware application in future.

3.1 TECHNOLOGY CHOICES

The programming language that I chose to develop this project was java. Why did I choose to use java? The choice of a programming language for this project was in one sense the most important choices to be made when doing the project. This choice would affect most of later implementation discussions in the project. There are several different programming languages that could have been chosen, including VB and C++. The selection of java above these other implementations came down to a number of different factors.

One on the main factors, in this selection, was that I am already very familiar with java and have a lot of experience developing projects of various different sizes using java. I also passed the Sun Java Programmers Certification Exam, SJPC. It therefore made some good sense to continue with a language that I knew rather than try to learn a new language from the beginning just for the project, the focus of the project is not the implementation for the sake of implementation but to try and get the best product possible developed in which to test the hypothesises in the shortest possible time.

Another factor that was in favour of the choice of java is that java is designed to be platform independent. This is a significant advantage in a distributed application. Writing the code for the project in a language that supports the notion of "Write once, run anywhere". Most distributed systems, almost by their very nature, have to live in a heterogeneous environment, where different systems could have different hardware architectures and could be running different operating systems. In this case, the project would be restricted to an operation field of a subset of computing platforms, compared to the java implementation, which can be used almost anywhere without significant alterations for each platform.

Earlier on in the discussion of the communication facilities in the project I outlined that the java language and the packages available for use contained good networking facilities and also has very good support for developing distributed applications in the form of implementation of Remote Method Invocation systems using java.rmi facility.

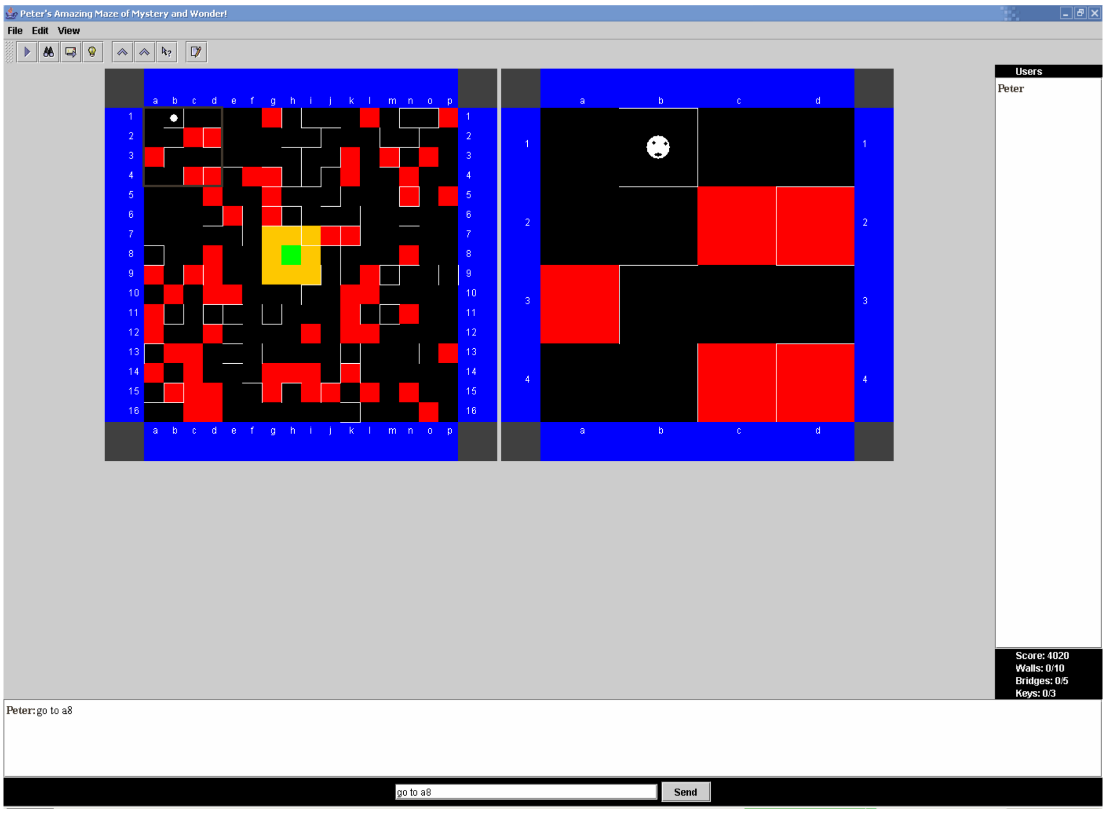
The java programming language also contained a useful system for designing graphical user interfaces. There are actually two such systems, javax.swing, Swing, and java.awt, awt. The awt packages are designed to allow programmers to develop heavyweight graphical user interfaces that should look the same on all system. The Swing packages are a lightweight solution, which is built on top of the awt system. The swing package has the advantage that it will allow the development of a GUI that can mimic the appearance of the host system on which the program resides.

This has the distinct advantage of permitting the developer to create a user interface that will be familiar to the user already in many respects. If we are trying to design a system that is trying to test the users interaction with other users using this game it was important to limit as much as possible the need to learn new user interface paradigms as I believe that this would effect the eventual results obtained from this study.

3.2 Graphical User interface

In this section I will outline the main elements in the user interface developed for the project. The program had to have a graphical user interface as it had to be able to support many advanced features that just could not be adequately be supported in a command line based system.

Now let us outline the basic design of the interface and then expand this with a few screen shots of the actual application.



Peter:go to a8

go to a8

FIGURE 2: GRAPHICAL USER INTERFACE.

As can be seen from the above figure the main interface contains a number of standard features you would expect from a graphical user interface. For example note the menu bar at the top of the screen with standard menus called "File", "Edit", and "View".

Users would have certain built-in expectations when they encounter these menus. For example the file menu should contain commands for starting or stopping the application or creating a new document for editing. The edit menu should group together commands for manipulating the document under inspection. The view menu should contain commands for manipulating the view presented to the user. For example the in a word processing tool this menu would provide for commands allowing “print preview” or “zoom” views of the document.

The commands in the view menu typically do not alter the document under consideration. Finally the help menu should contain commands that would allow the user to find information on how to user the system. These assumptions in effect represent a contract that must be respected when trying to design a graphically user interface.

In this system the file menu contains commands for joining and leaving a session and the view menu contains commands to display various different views to the users. For example there is a command here to turn on and off the radar view pictured to the right in the above figure. Another command here is one to turn on / off the view of telepointers in the system. A finally another view present is the ability to view the chat console or not.

The next standard interface widget that is often contained in a typical Graphical user interface is the toolbar. This should provide a shortcut or simple mechanism for accessing frequently used commands by pressing a button. In this program the toolbar contains buttons for adding or removing walls from the maze or adding bridges to the maze. Due to the nature of the game these commands should be used very frequently, therefore it would make good sense to locate these commands in this position, where users would naturally expect to find them.

The rest of the interface contains the main viewing area containing the game screen and a map or radar view of the entire game. This is located in the most prominent location on the screen as the users will be interacting with these two views with the greatest frequency and it will occupy most of their attention. This is the location a user would typically expect to find the view of the document they are working on in a document editor or word processor.

Located to the side and bottom of the screen are tools that will provide some awareness clues to the users but will not have to occupy their attention to anything like the same degree as the functions located towards the top of the screen like the toolbars, or in the centre of the screen where the main working area is located. These functions while important will then provide some useful information to the users but will they should be able to engage with it in a more passive manner.

One example of this passive information that will help the user to understand the system, and hopefully ease their interaction with the other users in the system, is the use of colour coded names at the side of the screen. This gives the user two pieces of information, one which users are currently logged into the game. It also gives the user information on which user is doing what as all actions in the system are colour coded according to the same scheme. For example if the client notices on the Map view, a radar box coloured Red in the upper left corner. He may know, for example that a particular user is looking at that particular section of the map.

A final component of the user interface noticeable on the figure is the status display. This section displays key information about the system that the users need in order to use the system usefully. An example of the information provided here includes the number of keys so far uncovered or the number of walls or bridges available for placing by the team, or the current score of the team.

Below is a diagrammatic depiction of the user interface that shows the various different sections to the display.

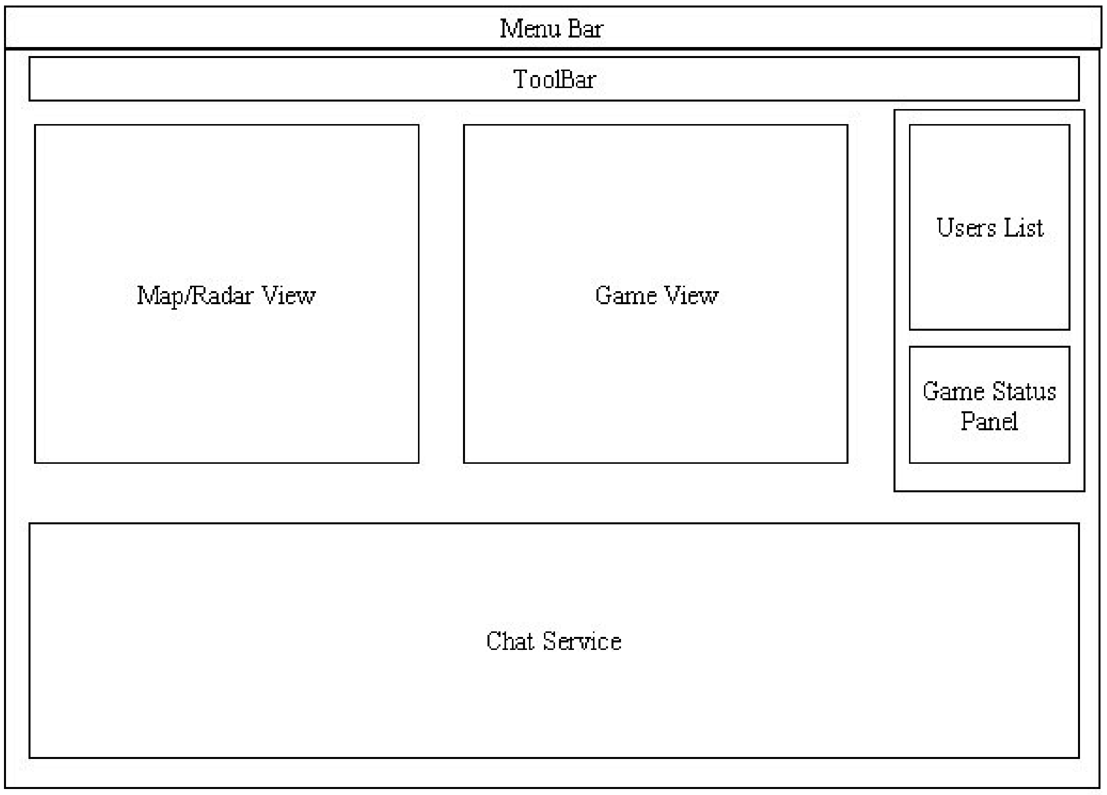
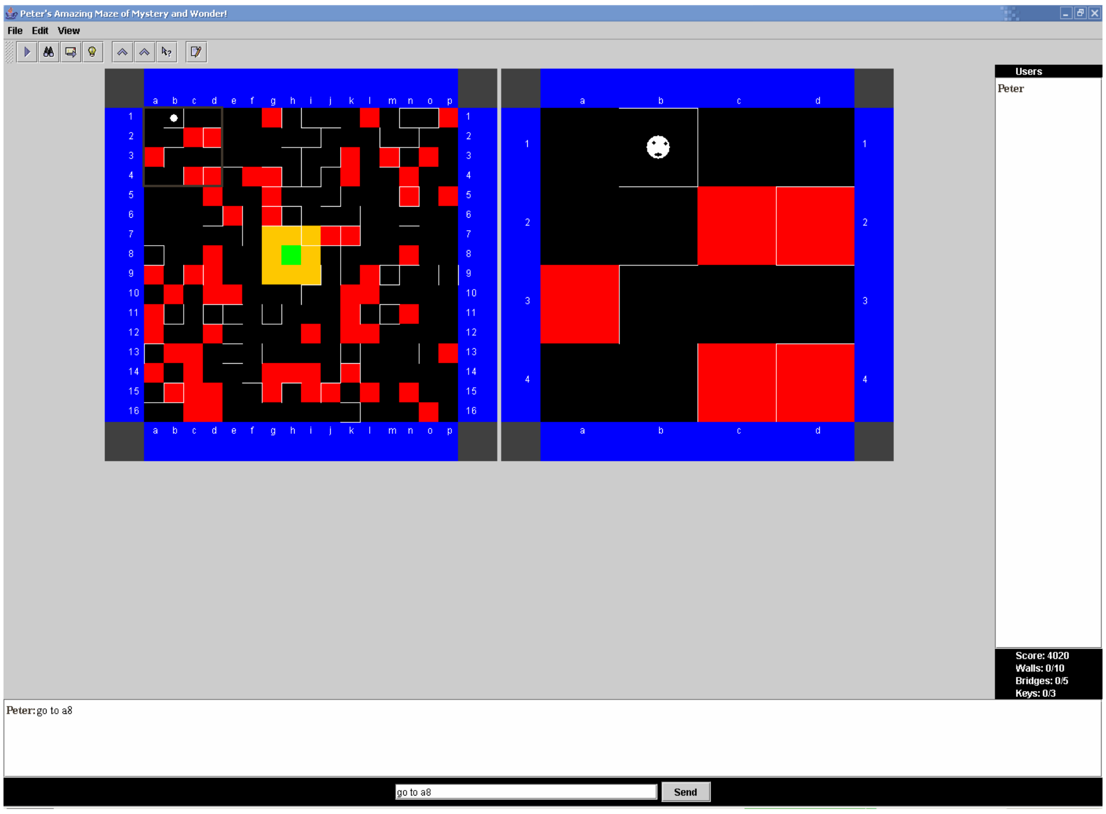


FIGURE 3: DIAGRAMMATIC VIEW OF GUI



Peter

Score: 4020
Walls: 0/10
Bridges: 0/5
Keys: 0/3

Peter:go to a8

go to a8

FIGURE 4: GUI SCREENSHOT

From the two figures above taken in conjunction one can see what the various elements of the Graphical user interface represent. Each element and its precise role will be outlined in greater detail in this section.

I will start by discussing the User List first as this is perhaps the simplest part of the system but in many ways is crucial for the understanding of the system and is one of the central Awareness components. This is simply a list of the users of the system; it was shown in [5] that it is important for users in a groupware system to be informed who the participants in the system are.

This is for a number of different reasons including certain psychological reasons. For example if the users know that one of the other participants is their boss they may experience certain inhibitions that could affect the outcome of the work. So depending on what type of user interaction you wish to support anonymity or explicit identification might be appropriate. In this case as we were trying to relate this study to collaborative writing, where all writers typically know who they are working with, using an anonymous system did not seem appropriate here.

One of the key considerations in a groupware system is how natural human interpersonal relationships and communication can be supported as transparently as possible. The central idea of this project was to draw an analogue between groupware and collaborative writing systems in general and the abstraction to system independent collaborative support.

In the system the names can be colour coded or not. It is hoped by colour coding the names and then using these colours consistently around the rest of the program, it should help the users get easy access to what other users in the system are doing. For example the radar views can be coloured the same as the users' names so that at a glance a user can tell where everybody else is looking.

The next section to be described is the chat system. During the study of computer supported collaborative writing it was noted on several occasions that it would be very useful if an application could support some form of informal communication. This was said to be useful for clarifying design issues and helping with conflict avoidance. The benefits of such system are also described in [25]. Therefore it was decided that it would be a very good idea to look further into this issue of out of channel communication.

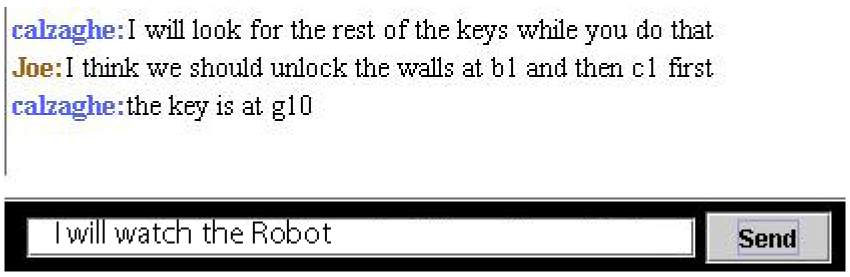
It is easier to avoid conflicting operations by improving communication that developing sophisticated algorithms for dealing with conflicts. Social protocols can prove to be a very effective mechanism for avoiding conflicting operations [24] and should be enabled if possible.

In [26] further support of integrated solutions is outlined. Here it is declared that there can be considerable performance improvements garnered if the context of a chat session can be efficiently related to the document. It would also be useful if the chat system fit in with the standard paradigms of text-base chatting so as to create a short learning curve and not to confuse the users on how to properly utilise the chat system. It has also be noted in [11] that the system should not be over constrained, so systems that deploy a limited vocabulary were ruled out, the users are then free to use the system as it suits them.

There are a lot of commercially available instant messaging solutions already available, Microsoft Instant Messenger and AOL Instant messenger, AIM, being the two most pre-eminent examples widely available. When deciding to include an instant messaging system in a groupware system there are several considerations. As mentioned in [26] it is helpful to have the IM client integrated into the groupware application as it can be viewed in context. Also by creating an integrated IM Client directly into the application you can be sure that all the participants in the study will be using a compatible system and that that they are all actually using it.

Most IM systems are topologically peer-to-peer systems, this is a useful feature for allowing large groups to separate into smaller working groups for particular tasks but has been noted in [24] that the system should multicast to all members of the session all the messages in the conversation. This is the mechanism that has been implemented here. Another feature of the chatting system is that all the participants' names can be highlighted in the same colours that are assigned to their names in the User list display.

The system records all the conversations conducted using the Chatting system, this can then be displayed as a webpage to be studied later to see how the participants actually used the system. The actual interface for the Chatting system is very simple and typical of almost all analogous systems. It consists of a display area for previous conversations and text area to enter text into the system. When new text is added it goes to the top of the display area and the previous messages scroll down.



calzaghe:I will look for the rest of the keys while you do that
Joe:I think we should unlock the walls at b1 and then c1 first
calzaghe:the key is at g10

I will watch the Robot

FIGURE 5: CHAT CONSOLE

The next part of the Graphical user interface to discuss is the rather simple display that is used to give the users some global information about some of the shared objects. In this display there are four properties that need to be reported to the users in order to play the game. This information is the current score by the team, the number of walls, and bridges, available and the number of keys found so far.

These elements will be described in the next section. This information is important because the user can use it to evaluate the performance of other members of the team without explicitly asking the other users. For example if one member of the team is assigned the role of finding the keys in the game he can just look at this display instead of having to continually ask them how they are doing. This is an example of feed through awareness in action.

This is akin to seeing the work count for various sections going up in a collaborative writing system so that you would be aware that the other users are actually doing some work. The score can be used in this study as one empirical measurement in that as more of the awareness mechanisms are introduced into the experimental scenario the users should be able to perform more efficiently and hence get a better score.

The two main heavy weight sections of the display need to be outlined next; these parts are the map/radar screen and the actual game screen. Each of these has a number of commonalities and therefore it makes sense to discuss them together. There are several components to these interfaces, including the map, the navigations buttons, and the location indicators as well as the actual iconography of the game itself.

The interface components are displayed below together with a diagrammatic view of the interfaces.

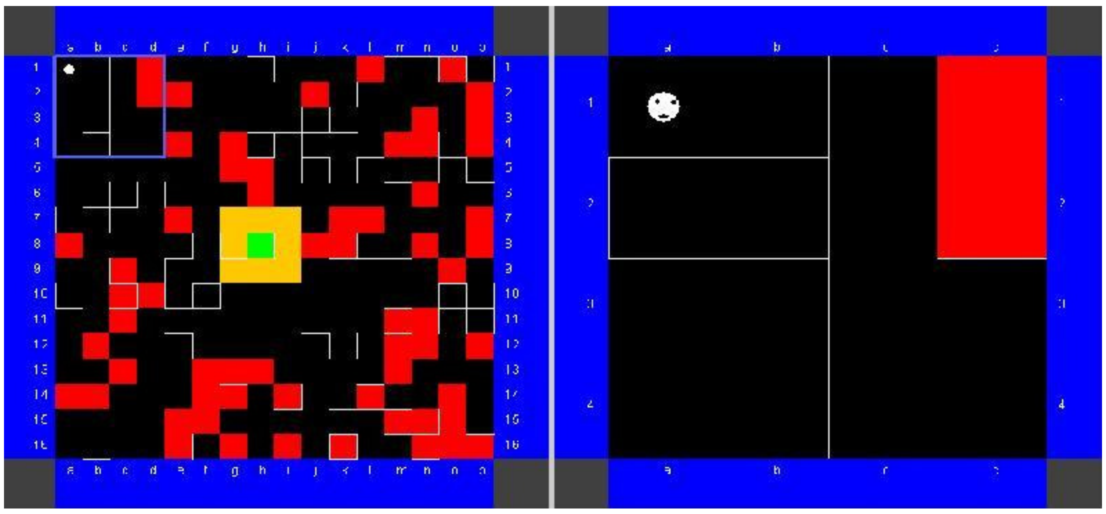


FIGURE 6: MAP/RADAR AND GAME SCREENS

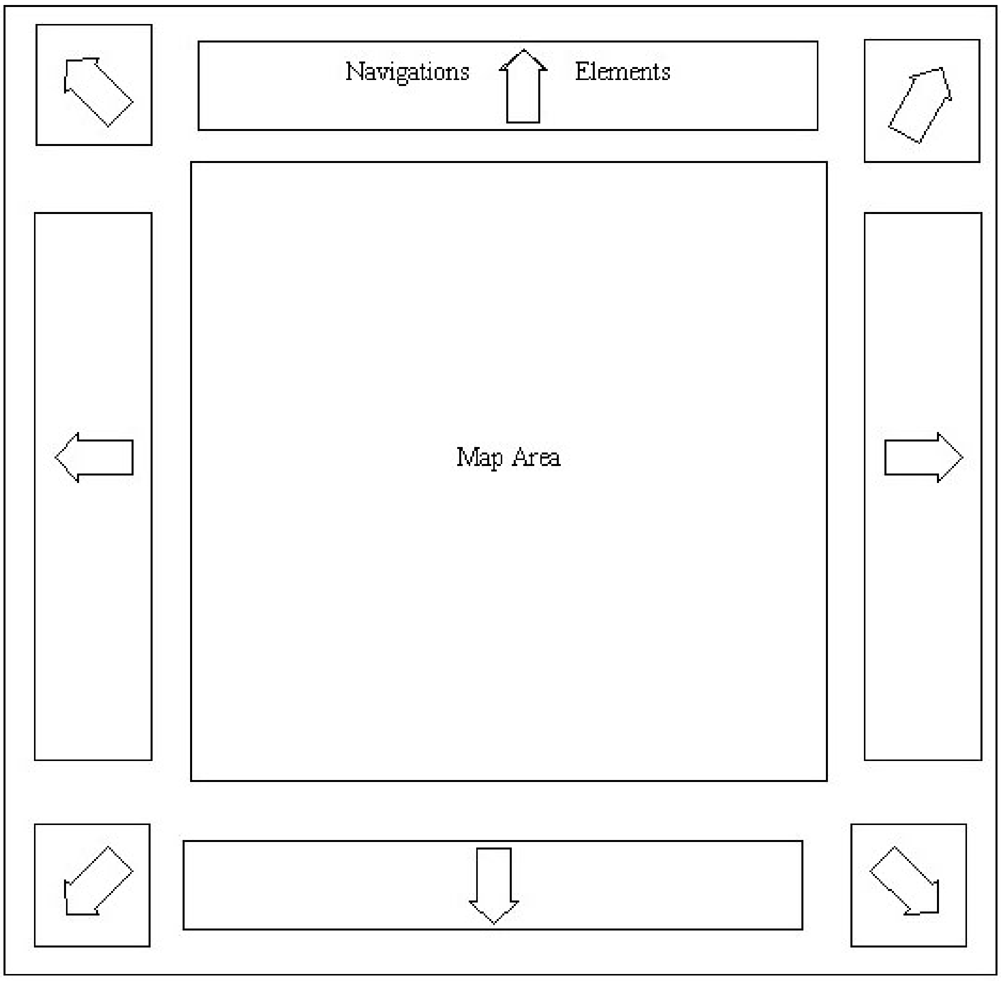


FIGURE 7: GAME SCREEN

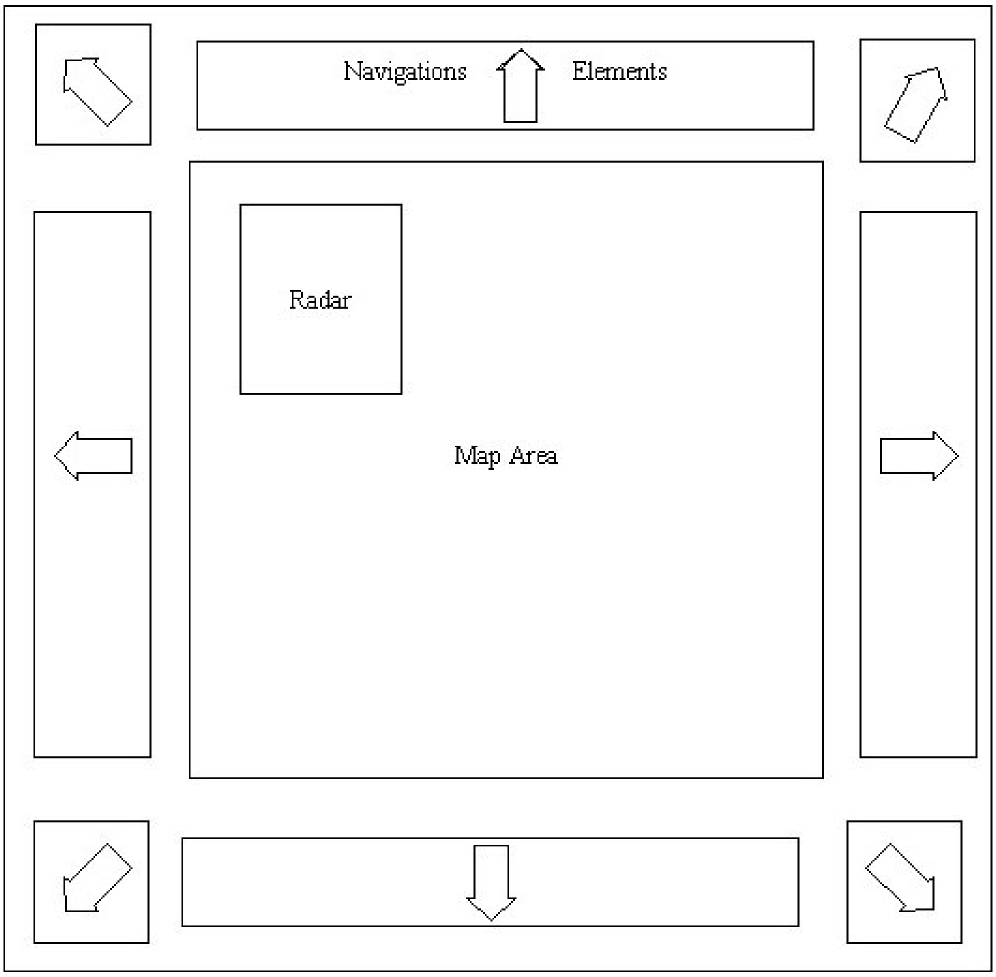


FIGURE 8: RADAR SCREEN

There is also a coordinate system in the program, not pictured in the diagram. It is similar to the system used in chess. Letters indicate columns across the top and bottom of the screens and numbers indicate the rows along the sides of the screens. The game screen is essentially a zoomed in version of the map screen. As the user navigates around the map the coordinates displayed along the sides of the game screen change accordingly to where on the board the user is currently viewing. The radar on the Left indicates what portion of the board the user is currently viewing.

The navigation elements along the edges of both screens serve the same purpose. The users can use them to move the currently viewed portion one place in the desired direction. The user can move the view in the following directions, North, North-East, East, South-East, South, South-West, West, and North-West.

By using the Map screen the user can move to a random location on the map, this gives the user direct access to all portions of the map.

Different information is displayed on both screens. The Maze that is displayed on both screens is the same data, but it is interpreted in a different manner on both screens. In the radar screen for example, the entire board is displayed, where only a portion is displayed on the game screen. Certain objects in the game are only displayed in the game screen, for example, the keys.

While playing the game most of the interaction is performed through the game screen. It is here that the users can add and remove walls from the maze and add bridges over the traps. It is also here where the players can collect the keys. The keys are not displayed in the map screen as then finding them would be the most trivial exercise and would eliminate the point of having them in the system.

The robot player is shown in both screens. It is the job of the players in the system to ensure the safety of this robot and guide him to the finish. The robot discussed in more detail has its own maze solving algorithm and should find the finish on his own provided that he does not fall into the traps, which he has no knowledge of and provided that there is currently a root that can be taken to the finish. At the start of the game the finish is surrounded with a series of traps that can only be overcome by finding all the keys.

Now that the Graphical user interface should be quite clear, it would be appropriate to discuss the actual architecture of the system and describe how this system actually works.

3.3 Design Architecture

As mentioned earlier the programming language used for building this tool was Java. Other key elements to the system are that it is going to have to support multiple users synchronously; the result of this is that the system must be able to work on a network to allow users to access it from different terminals.

The fact that it will be working on a network results in a number of built-in problems from the start with the implication that all communication is through messaging. There are a number of classic problems that you are going to encounter when using networked systems, for example, the lack of a global clock system for ordering all actions and no global knowledge, in that it is impossible to know for certain the current state of other systems on the network. A final problem is that of partial failure. This will be encountered when users systems fail but this is not known to the system as a whole.

The goal of the project was to create a system or framework in which awareness mechanisms can be reasonably tested. It was not to solve basic network and distributed systems problems from first principles. It was with this in mind that I addressed the considerations in the preceding paragraph. The first issue of a lack of a global clock was the most pressing and had to be explicitly addressed in the project.

The issue of partial failure was in most respects ignored as the system was to be deployed on a LAN with good reliability and throughput etc. and could be assumed to work most of the time or at least for long enough to test the system. The issue of lack of global knowledge was important as all the clients should have the same view of the workspaces, or more correctly should have the same workspace objects, the actual view of the data can be different for each client and this should be explicitly supported.

The lack of a global clock was addressed with consideration to [9] this utilised a vector time stamping mechanism. All actions in the system were broadcast to the other users in the distributed systems and ordered using the vector timestamps.

Most of the basic networking issues could be sidestepped by the use of the Remote Method Invocation implementation in java. This provides a transparent means of contacting the other clients and updating their state. It allowed quite sophisticated interaction between clients. For example Objects could be passed between clients rather that trying to create bespoke packet or messaging formats for communication between clients.

The main topology of the systems was to be a star-mesh topology. This topology was chosen for the efficiency of setting up the experiments and to enable more efficient collection of results. It could have been set-up as a pure mesh, or peer-to-peer system if necessary but this was considered to be impractical as there would be more overhead involved with starting new games for example and other bootstrapping operations to get the system up and running each time. In such a peer-to-peer system if for example you wanted to create a new game, who would create it and then how would the others join the game? These are not particularly difficult problems to solve but it would involve a lot more coding overhead for features that would not actually add to the core functionality that we wish to test.

By using a central, known, server for the creating the game and controlling all automatic actions in the system it was possible to simplify the actual clients presented to the end users. The server was used for discovery and setup; it could also be used as a monitor of the system as it received all communication from the peer clients as well. They would not need to know anything about creating or destroying games or sessions. That would have been an unnecessary on the end-user in this context; all the users have to do to get everything working is enter their name at the login prompt.

The diagram following should explain the star-mesh concept in a clearer fashion. The central server deals with managing the users and the session and naming and locating of all clients in the systems. It also creates all the games and handles a lot of the updating to the users. All actions that subsequently affect the game board after the session has been created no longer go through the server but are propagated in a peer-to-peer manner; this is a must as if all actions went directly through the server the server could impose an ordering on the actions and then distribute them to the clients in an ordered way.

A client/server program can be represented using a peer-to-peer mechanism but the flipside is not necessarily true. If I allowed the server to control all interactions then the system would not be as general as if I designed a peer-to-peer system. The server was simple a convenience for the mundane tasks that were beyond the scope of the project.

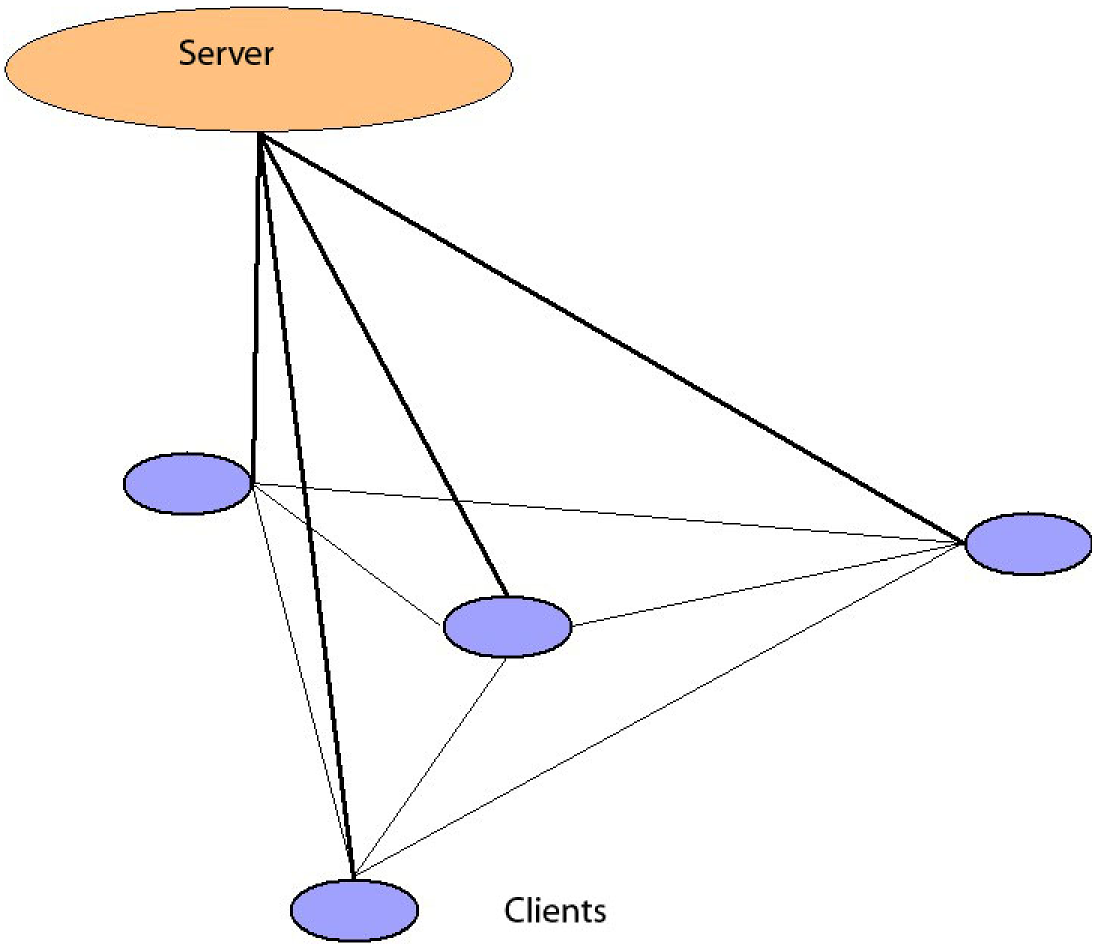


FIGURE 9: TOPOLOGY

When the server is first started up it creates a new board that will be configured according to the aims of the particular test scenario that is to be run. Then as each of the clients log into the system they will be presented with clients that are configured also according to the same settings. In this way all clients should have the same user interface and should be looking at the same board.

As the game proceeds and the users make changes to the game environment the changes are sent to the other clients in a peer-to-peer fashion. Each client then transforms its local copy of the game in the same manner so that at the end of the game all clients should still have the same data to view.

As each client logs into the system the server is able to get its IP address. The server records the names and IP addresses of the clients along with some other information to do with clients for example the colour used in the system to represent the client in the game interface. This list of users is sent to each client that the server knows about every time somebody else logs in; this means that each client will then know the IP address of all the other clients in the system. This information is necessary for the communication to continue on a peer-to-peer fashion for the rest of the session.

Armed with this information the clients, as they make changes to the game board broadcast the changes to the other clients that they know about along with the server, the server acting now as any other client in the system, so that all clients can now maintain a consistent data set. The client also needs to be informed of the changes to allow new clients that log into the system later to get an up-to-date copy of the data without having to alter any of the clients to deal with this. Late joining clients can join the system transparently in this way. There is no need for a special protocol to update the new client it is brought right up to speed as if it joined from the start.

All communication in the system is over RMI; this is a very fully featured communication mechanism. The key advantage of using RMI rather that a message passing system is that it is cleaner to implement and clearer conceptually. With RMI you are able to take advantage of the power of Object-Orientated programming fully rather that relying on some proprietary system. In a strict message passing system the interaction is more like the first part of the next figure, where the RMI approach is more like the second.

The message based system has one advantage, and that is that all communication goes through a central point where all actions could be for example be ordered centrally rather that the more cross-cutting code you get with having to check this in several different places in the RMI based systems.

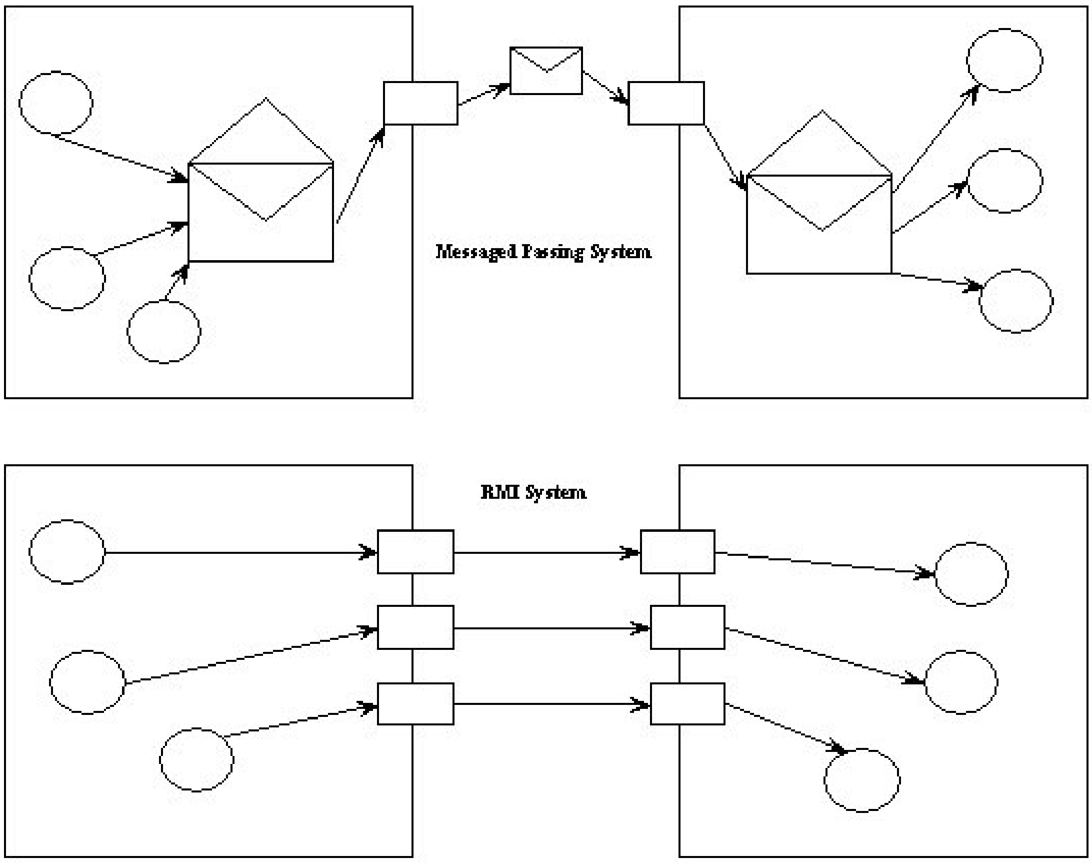


FIGURE 10: RMI VS. MESSAGE SYSTEM

By using the message based communications mechanism you would in effect have to create your own protocol for communication, you could achieve almost everything you can with RMI but it would be a less flexible and extensible system. RMI allows using java's reflections mechanisms and mobile code, with dynamic class downloading and loading. This would allow the system to be more flexible.

RMI would also support call-back functions relatively simply. The only disadvantage to RMI was ordering incoming actions. This was tackled by using a global stack that all interface methods would access to get there actions ordered. For the above reasons it was decided to go with the RMI approach. A closer look at the actual implementation would be appropriate now.

3.4 PRODUCT DETAILS

In this section it is hoped to show some of the finer details of the tool developed. Beginning describing the algorithm and data structures used to order the actions at the clients, then moving on to follow-up with a general description of the program and the key communication algorithms. Next a description of the data representation of the actual game and graphics is called for, finally an outline of the main awareness systems in the tool will be illustrated, also presented will be general problems and key design decisions.

3.5 ORDERING ALGORITHM

All actions in the system need to be ordered to occur in the same order at each site but also occur in a timely fashion. These are contrary requirements, as local actions, in order to appear responsive should happen immediately and then be sent, but other actions at other sites could be happening in parallel or even before this current action. This is where conflicts arise, to over come this there are several solutions available, [1], [2], [3] describe several algorithms for automatically maintaining consistency when working in distributed systems, they are focused on algorithms for operational transformation on tree based data-structures that would be typically used to represent documents now, you just have to think of XML, XHTML, SGML, etc. they can be used on flat data-structures with a few adaptations, they also require an algorithm for dealing with conflicts.

The simplest change to make is to remove the overlaid tree data structure and compare the timestamps of all actions as they occur. What I mean by this is when using a tree based operational Transformation algorithm the algorithm are effectively partitioning the set of actions according to their location in the tree and then applying conflict resolution algorithms recursively on conflicting sections of the document.

In the flat, sequential system, all actions are in some senses conflicting and only have to be undone if they occurred logically before another action and then this new action is done, and finally redo the original actions after this new action. This undo/redo scheme can be best described by using an example using some time stamp values.

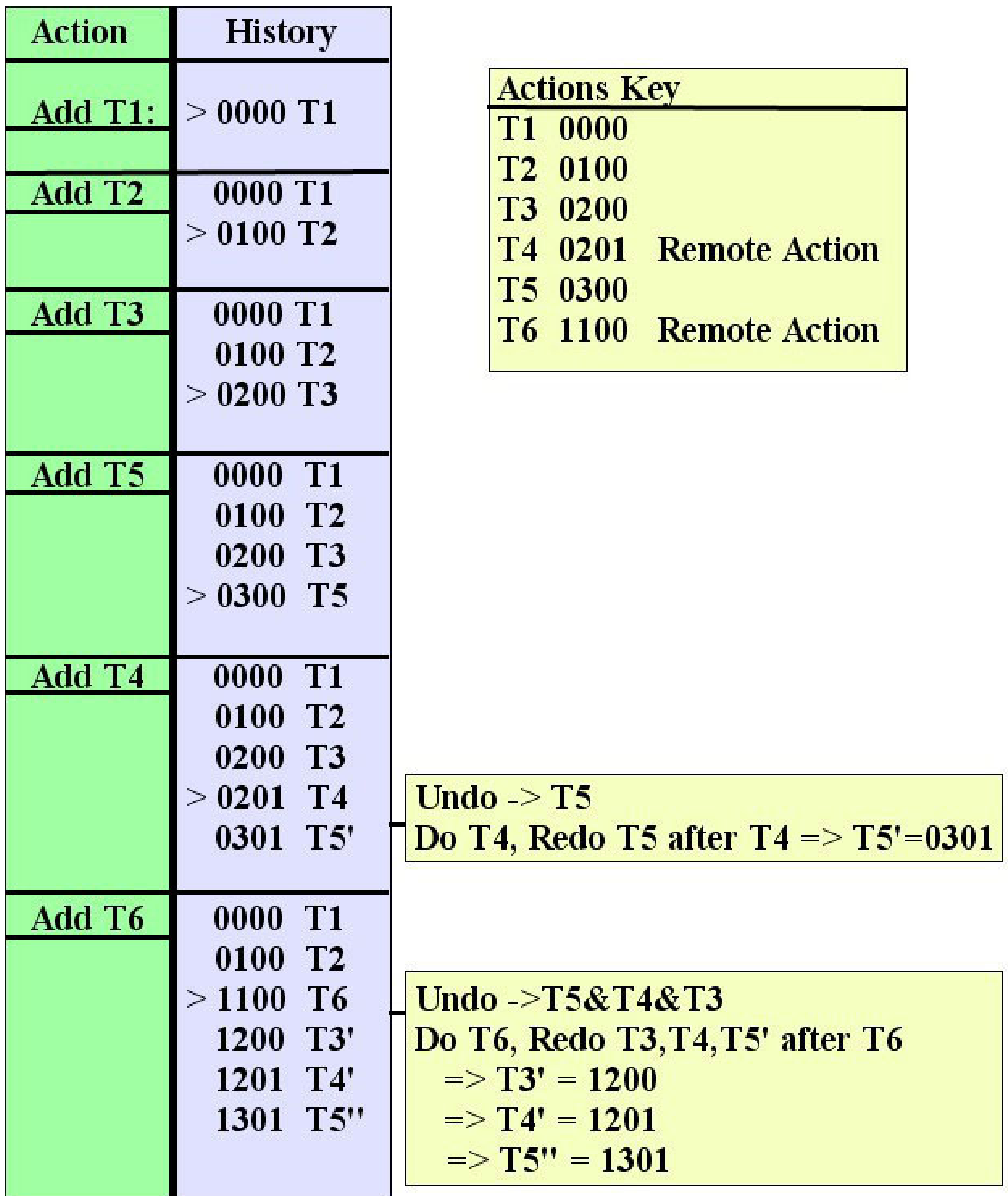


FIGURE 11: VECTOR TIMESTAMP HISTORY

As should be clear from the figure, actions that arrive at the client after an existing action has occurred must be undone, then the action that arrives is performed, finally all the actions that occurred after it must be redone, but importantly they must account for the change in context necessitated by a new actions having occurred before it now.

The vector time stamping was based on the systems of logical clocks in [9]. The algorithms for merging and evaluating the order of the vector timestamps are also from [9]. I have implemented these algorithms and the system is able to use these timestamps correctly. At the moment the ordering system has not been applied to many of the actions in the system yet, but the core functionality for this ability is built and can be used.

3.6 PUBLIC INTERFACES

As the system is going to be a distributed system that would live in a network environment, and also because it would be a peer-to-peer system where every instance is essentially a client and a server, I think it was necessary to keep a clear distinction between the public and private interface of the system. This was best practice to keep the system clear.

The server would export an interface that would allow the clients to log in and to get copies of the game. The client would export an interface to update the current state of the system but also mirror some of the interface methods of the server to allow the server to easily call back the clients with updated information. A prime example of this is when a new client logs into the system, the server has to be able to inform all the clients of this new user in the system. The server, as it is to track all the changes made to the system by the clients as well, for reasons mentioned earlier, must export the same interface as the clients.

Every element in the system, a client or a server has an rmi registry running on it. The server actually has two rmi registries running on it because the server only acts as a server when a client logs into the system but all the rest of the time it is really just another client in the system, albeit a silent client with no interface. Therefore the server interface is in effect a superset of the client interface.

Typical methods in the client interface are editBoard, addusers, chat, moveMan, and moveFocus. Typical methods in the server are loginUser, logoutUser, getBoard, getMan, newMan, as well as the client methods. All are fairly self explanatory, or will be discussed in more detail later where appropriate.

3.7 PACKAGE DIAGRAM

The program itself is quite long, it is therefore completely necessary to put some structure on this complexity. The program can be decomposed into several distinct elements, for example, separating the roles of client of server can prove useful, even if all clients are both clients and server simultaneously, the server is both a server twice and a client in the system. Presented below is a package diagram followed by a rational for this particular decomposition.

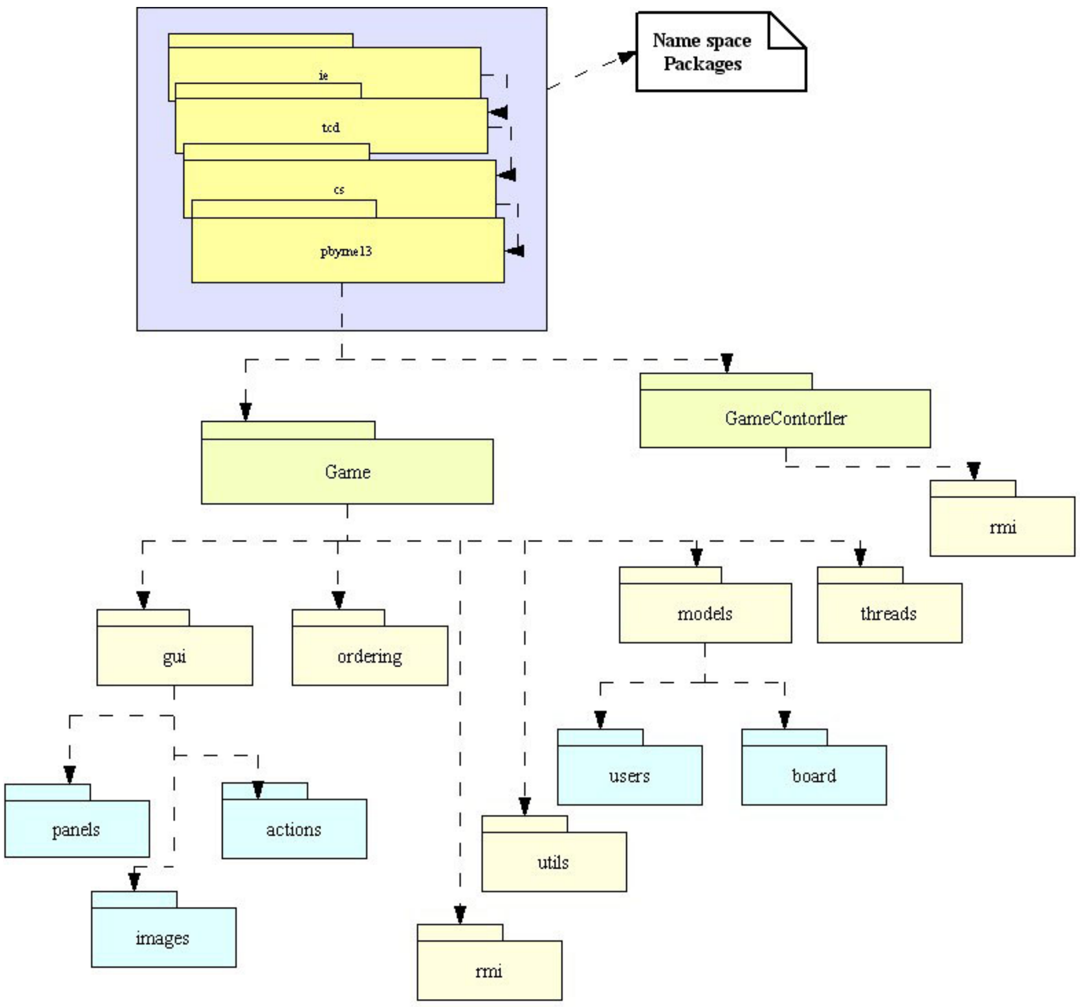


FIGURE 12: PACKAGE DIAGRAM

The first four packages of the system are there for naming reasons. They are to insure that we are using a distinct and unique namespace for the project. It is very unlikely, using best practice that anybody else will have packages starting ie.tcd.cs.pbyrne13. The next two, essentially top level packages are, game, which contains all the packages to needed by the clients for the GUI, data representation and control, and gamecontroller, this is a package that contains all the external code for the server. A lot of the server code is in the game package as the server needs a lot of the same objects and is also essentially a client. That is why the distinction is not strictly between client and server at this level.

If we drop into the game package there are six packages in the first layer. The function of most should be fairly self-explanatory for example the GUI package and its children are for managing all the elements of the GUI and user interface events.

The ordering package is responsible for all of the ordering and vector timestamp management functions. The utils packages contain some miscellaneous packages for routine tasks and debugging etc. The threads class contains classes for controlling a thread that manages the lifecycle of the robot that navigates around the system. The action is performed by the server as it is the only place it makes sense to put it, as other wise you would have to arbitrarily choose a client to control the robot. This would not be a good idea as I wanted all clients to be identically configured in the systems so that the end-users would not need to know anything about setting up a session.

The rmi package in the game package deals with the public interface of the clients, classes that export a remote interface are stored here. The remote proxy design pattern for rmi was also used, and the proxy classes were also created here. These proxy classes were used so that client call remote objects exactly like local ones and would not have to worry about remote exceptions or other elements to the actual networking; it could all be contained in this wrapper.

A client would get a remote reference to other clients or the server through these proxies and could call any of the remote methods of the system they are calling with any consideration of rmi. The rmi package in the gamecontroller performs an identical function for the server program.

The models package contains two child packages, board, and user. The first package "board", contains all the classes necessary for managing the data-structures that represent the board and other elements in the game, they are fundamental to the entire tool and will be discussed in particular in the following section. The "users" package contains the classes for the representation of users in the system.

3.8 Representation of Board, Cell, Wall

Now that we have looked at the main structural breakdown of the program itself it is time to look in more detail at some of the key classes in the system. The first thing to look at is the representation of data in the system. Starting by looking at how the board is represented, followed by the cells and finally then walls in the system.

The board is essentially a wrapper for two data-structures an ArrayList and a two dimensional array, this data structure is almost exactly the same as a Vector but it has less overhead relating to synchronised access to its methods. The constructor when called creates a new board with all the cells and walls randomly assigned and configured. There are public methods for getting access to these data structures. You can imagine the game board as two overlaid structures, the underlying structure is the ArrayList that contains all the cells or squares on the board. The second overlaid structure is the Array of walls in the board.

The cell class is an object for representing a square on the game board. The cell has various different attributes, including, whether or not it is safe to walk on, if it is the start or the finish, does it contain a key, or a door.

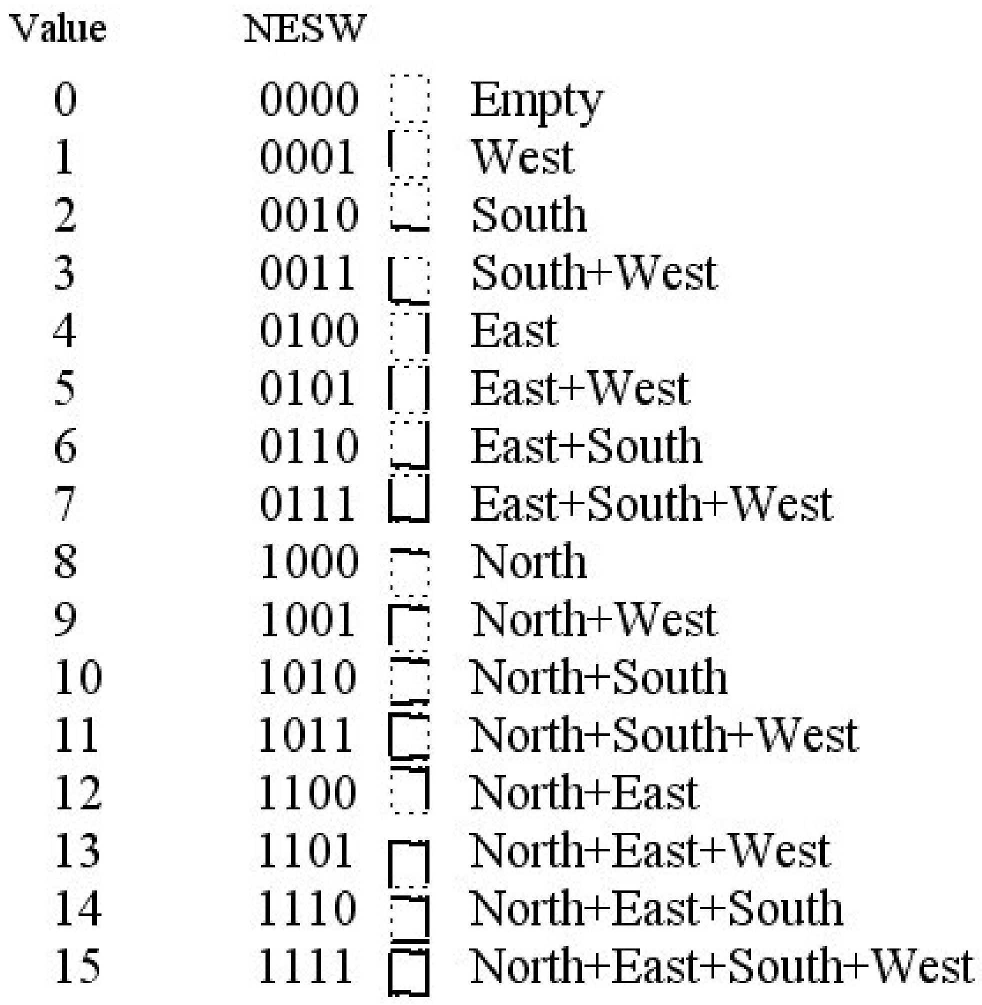
All cells have two other important attributes, x and y coordinates for the system. It was necessary over-write the “equals” method of the cell class, two cells are equal if they have the same x and y coordinates. There should be only one cell of each combination of x and y coordinates in the board.

These cells could have been stored in a HashSet data structure as this would guarantee that the cells were unique and you would only be able to add one of each coordinate. To change this, the way the code is implemented, it would be possible to change to this data structure by doing a find and replace for the ArrayList declarations etc in the java files. The HashSet was not used in this system as it would require over-writing the hashCode method, which would use some hashing algorithm to generate an appropriate hash of the object. It was felt that this was not really necessary and by building in restrictions at this level you would be restricting the extensibility of the code in future, the checking could be done at another level, like in the code, easily enough.

The representation of the wall is simply an integer. The last four binary digits of the binary representation of the integer are used as flags to indicate what sides contain walls. The system can then find out which sides contain walls by shifting in turn 0, 1, 2, or 3 places to and then “anding” it with 1. If this returns 1 then that side contains a wall. It is possible that if one side contains a wall to the south, and the cell to the south contains a wall to the north then you would have a double wall. It was then necessary to check for all these double walls and remove them.

Whenever the system needed to add a wall or check if a wall was present it had to check both cells to make sure there was a wall or not on that side. The below figure should clarify this representation.

This representation was very efficient and was much easier to deal with than creating a composite object for the walls or including it in the cell class. The code for checking for double walls and the like would be very slow in this case, as you would have to find the cells you wanted to look at in the ArrayList and then get the set of walls for that cell and then find the other cells you were interested in and compare the walls from the two cells. This would have to be done for every cell every time the screen is redrawn, it would have slowed down the drawing to the screen, presenting a clunky interface to the end user.



Value	NESW	
0	0000	Empty
1	0001	West
2	0010	South
3	0011	South+West
4	0100	East
5	0101	East+West
6	0110	East+South
7	0111	East+South+West
8	1000	North
9	1001	North+West
10	1010	North+South
11	1011	North+South+West
12	1100	North+East
13	1101	North+East+West
14	1110	North+East+South
15	1111	North+East+South+West

FIGURE 13: WALL REPRESENTATION

3.9 BACKTRACKING

In the game there is what can be described as a robot player, which navigates the maze and tries to find the finish automatically. The objective of the game was to ensure that as many of the robots could find the finish safely and as quickly as possible. The robot should follow some simple heuristic that will allow the players judge where he is going to move to next. The robot should account for certain obstacles in the game while moving about the board, and should also be aware to changes made to the board as he navigates his way around the board.

The algorithm that we decided to implement to perform this searching of the workspace was a simple backtracking algorithm using a stack to control a history of moves. As the robot moves to a new location, it pushes the coordinates of the cell onto this history stack. Then when he reaches a dead end with no where left to go, he pops the last location off the stack and tries any untried routes left from this location. If there is still a dead-end he keeps popping from the stack until he reaches a location from where he can try a new route. The robot cannot visit any cells that are in his history stack.

The robot only takes account of walls in the map not whether a cell is safe or not to walk on; it will walk onto unsafe cells, this will kill the robot and another has to be created by the server. The location of the robot is controlled by the server through the use of a controlling thread, when the robot is killed the tread is stopped and then another is created to control the new replacement robot.

The map is constantly changing as the players add and remove walls or add bridges to the map. Players also find keys, which will unblock the path to the finish. This being the case it is necessary to clear the history of the robot each time a change is made to the board as he could have ruled out a route that has subsequently become available.

By using this algorithm, then provided there is a path to the finish the robot will always finish as it is effectively a depth-first search of the workspace. This is an important guarantee as if it could not be relied to go towards the finish the game would only end by chance. This would reduce the effectiveness of the study as the time taken to complete the task would not be relevant any longer. The time to complete the task would not be related to effectiveness of the collaboration but more down to chance.

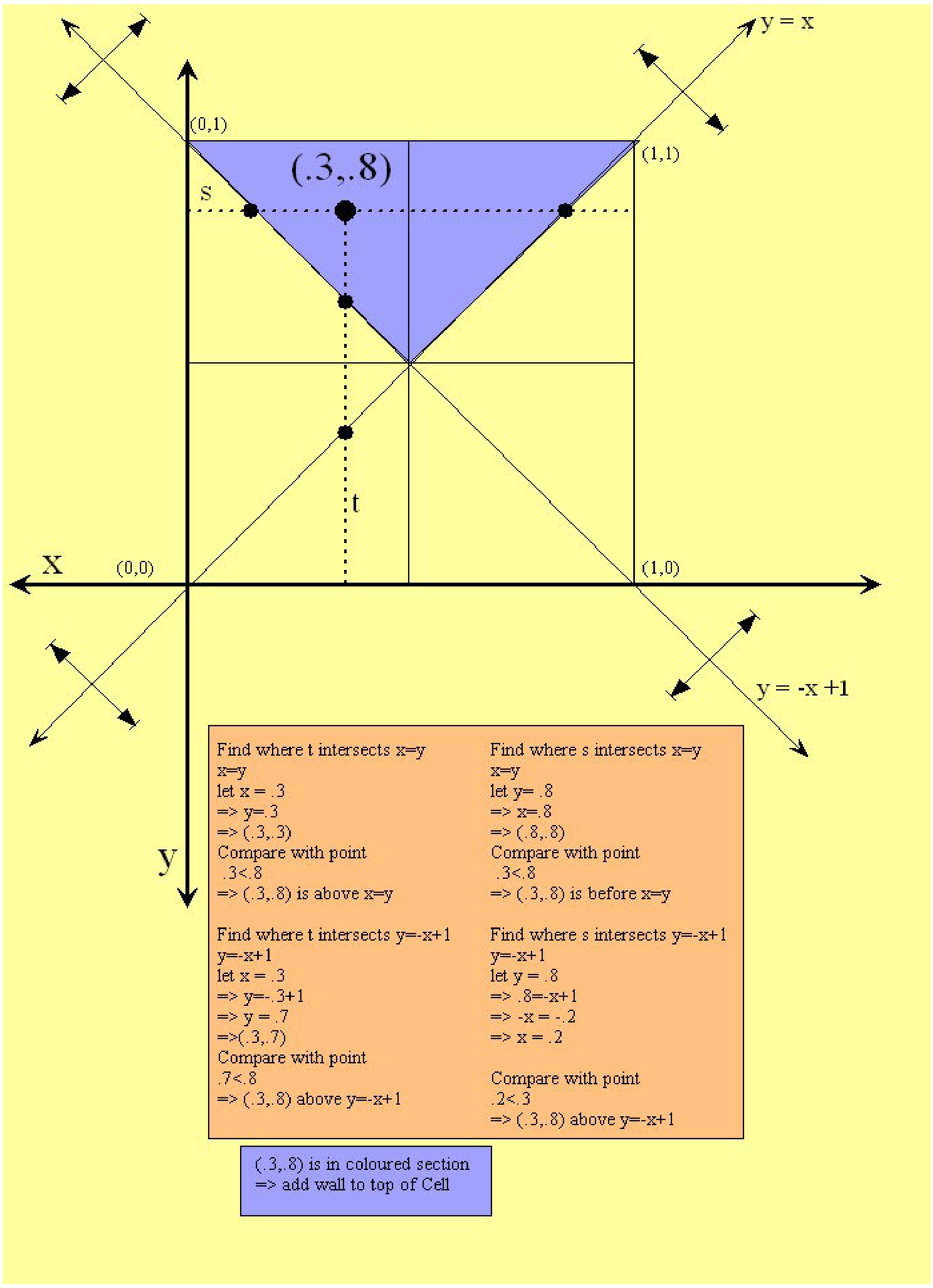
3.10 Radar, Telepointer and Ruler

It is possible to capture the mouse coordinates of any mouse clicks in the painting regions where the map screen and game screen are displayed. Using this and some arithmetic it is possible to convert from the x, y coordinates into the x, y coordinates of the cells and find out the actual cell that was clicked on. Using this information it is possible to send what portion of the board the user is currently viewing to the other clients in the system. They can take this information and easily work out where to draw the radar for that user in that users colour.

The telepointer is implemented in a similar manner but instead of rounding the coordinates to the nearest cell, the original coordinates are sent as this is required to be more precise.

A lot of the difficulty of implementing the graphics for the project was in the translation of the coordinates from the coordinate system used in the graphics context of java.awt and the representation of the board in the Board, Cell and Wall classes. There were conversions of this nature necessary in almost every action in the system. For example the ruler on the game view had to be constantly updated to reflect the current location on the board.

When adding a wall to the system, deciding which side it should be place involved finding the bounds of the particular cell the pointer was currently moving over, then finding the equations that would represent the diagonal lines of the cell. Then normalize the coordinates to a unit square. Then find whether the pointer was on each diagonal or above or below. It was then possible by comparing the point's location in relation to both diagonals to find accurately which side of the cell the mouse was currently located.



Find where t intersects x=y
x=y
let x = .3
=> y=.3
=> (.3,.3)
Compare with point
.3<.8
=> (.3,.8) is above x=y

Find where t intersects y=-x+1
y=-x+1
let x = .3
=> y=-.3+1
=> y = .7
=>(.3,.7)
Compare with point
.7<.8
=> (.3,.8) above y=-x+1

Find where s intersects x=y
x=y
let y= .8
=> x=.8
=> (.8,.8)
Compare with point
.3<.8
=> (.3,.8) is before x=y

Find where s intersects y=-x+1
y=-x+1
let y = .8
=> .8=-x+1
=> -x = -.2
=> x = .2

Compare with point
.2<.3
=> (.3,.8) above y=-x+1

(.3,.8) is in coloured section
=> add wall to top of Cell

FIGURE 14: PLACING WALLS

4 EVALUATION

4.1 EVALUATION OUTLINE

The application was tested in a small pilot study involving two to four users playing the game. The system as mentioned earlier is a testing platform for awareness mechanisms. The system would allow various awareness mechanisms to be turned on an off to test how it affected the users performance in a simple collaborative task. Outlined below are the various testing scenarios considered and a summary of how the users got on under the various different schemes. During the pilot testing process five different scenarios were used in the evaluation of the system.

4.2 THE TESTS

In this section it will give an outline of the various tests that were carried out as part of the pilot project. It will also conclude with a description of several different tests that would be useful to carry out as part of a larger trial. Finally it will describe some appropriate methodology for carrying out these experiments in future as part of the larger trials. The system is a testing platform for awareness schemes, there is also a map editor provided, so researchers can setup particular maps or they can edit the system variables. These can be saved to a file and reused. This will allow researchers in the future to run repeatable experiments. In these simple tests the maps generated were created randomly.

4.2.1 Without Awareness Assistance

The first scenario we tested was how the users interacted with the system when there were no awareness mechanisms on at all. The users had no communications channel available. This was to be the base case on which to compare the other results. The users had difficulty completing the task and only successfully completed it once.

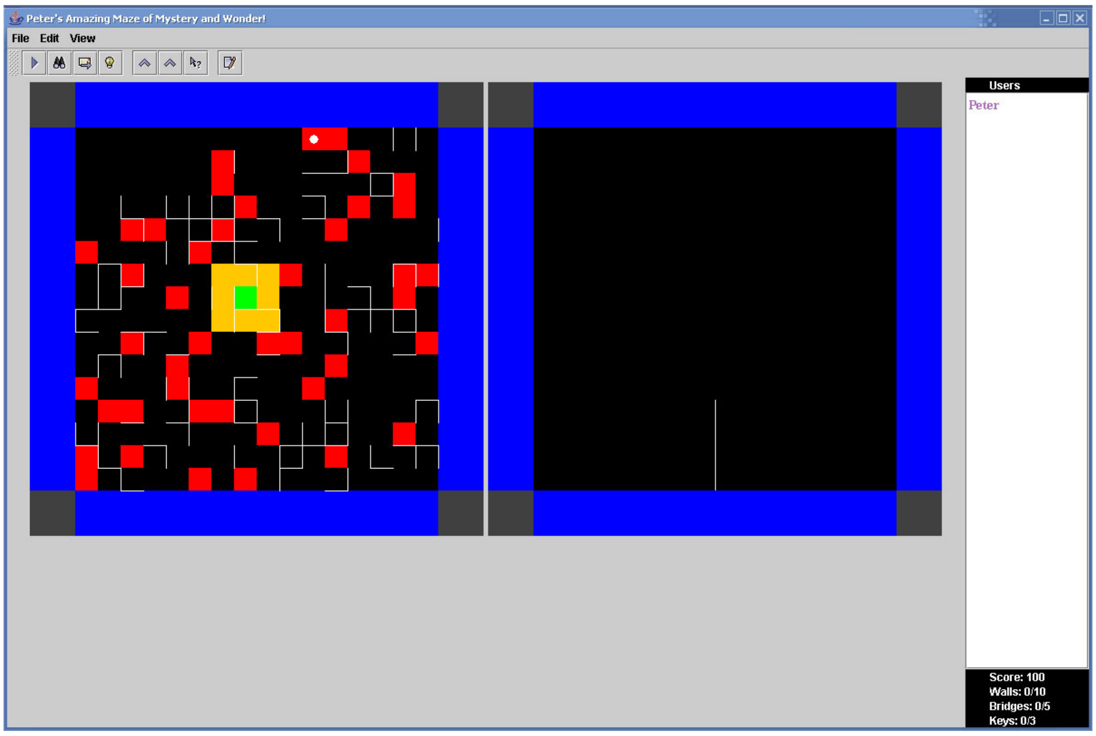


Figure 15: No Awareness Schemes

Then the next scenario the users were allowed communicate but still without the awareness schemes activated. In this scenario the users finished the task on all three occasions; the users were able to partition the job between them better. In this run one user would decide to look for the keys while the others looked after the robot. During discussions the uses would describe parts of the workspace with phrases along the lines of "add some walls in the section at the left near the middle, beside the two red squares".

4.2.2 Using the Radar

The next test performed was the use of the radar. In this scenario the radar was enabled. This allowed the users to know where other users in the system were. Again this test was run with and without communication. This time the users performed better in the scenario with out the communication mechanism than in the previous test. I think this is because the users where able to infer from the movement of the radars what the other users were doing. For example if one of the radars is seen to be scanning around the workspace they are probably looking for the keys.

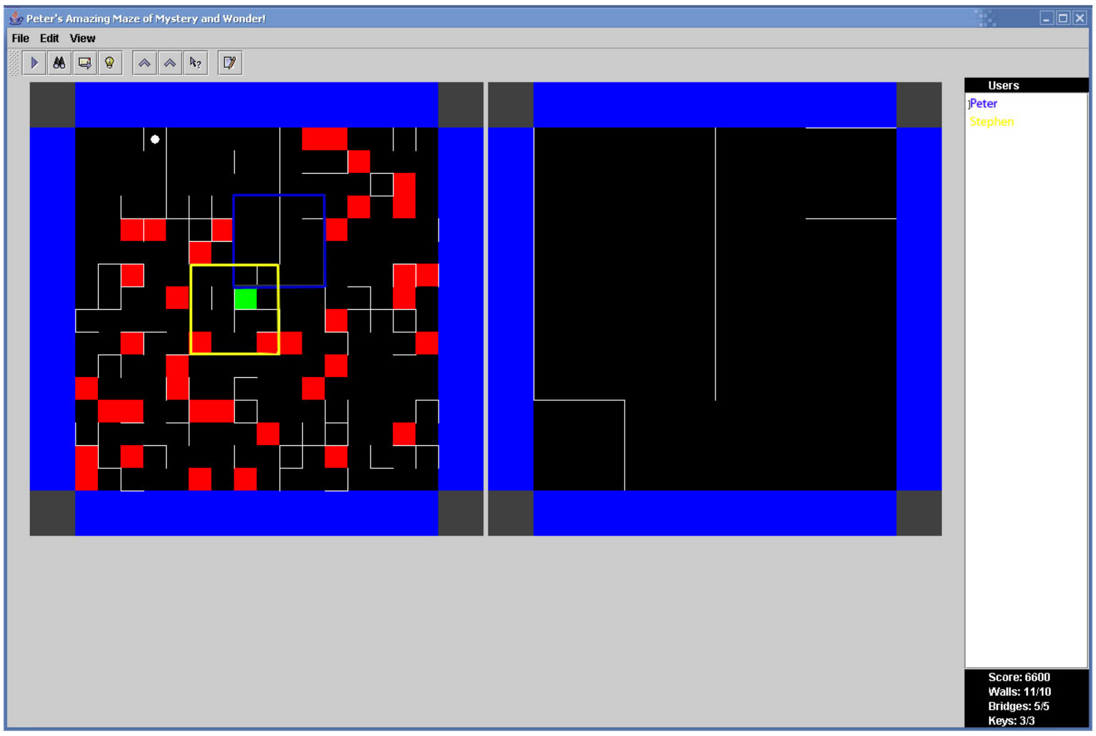


Figure 16: Radar View

The next test performed was the use of the coordinate system for referencing locations in the workspace. The users preferred the use of reference system in communication. It proved helpful to describe their location, and it could be expressed with simple notation. For example they could say G4, where before, they would have to say more like near the middle.

As can be seen from the image below there is a grid reference system along the edges of the game area. This can provide common reference points in the workspace that all the users can use for discussing the workspace. The system is based on the English Algebraic coordinate system used on chess boards. The image has been edited to highlight the ruler system, the top and left have been highlighted; the bottom and right have been left alone. The section highlighted in green represents the values that should be displayed in the right-hand game screen.

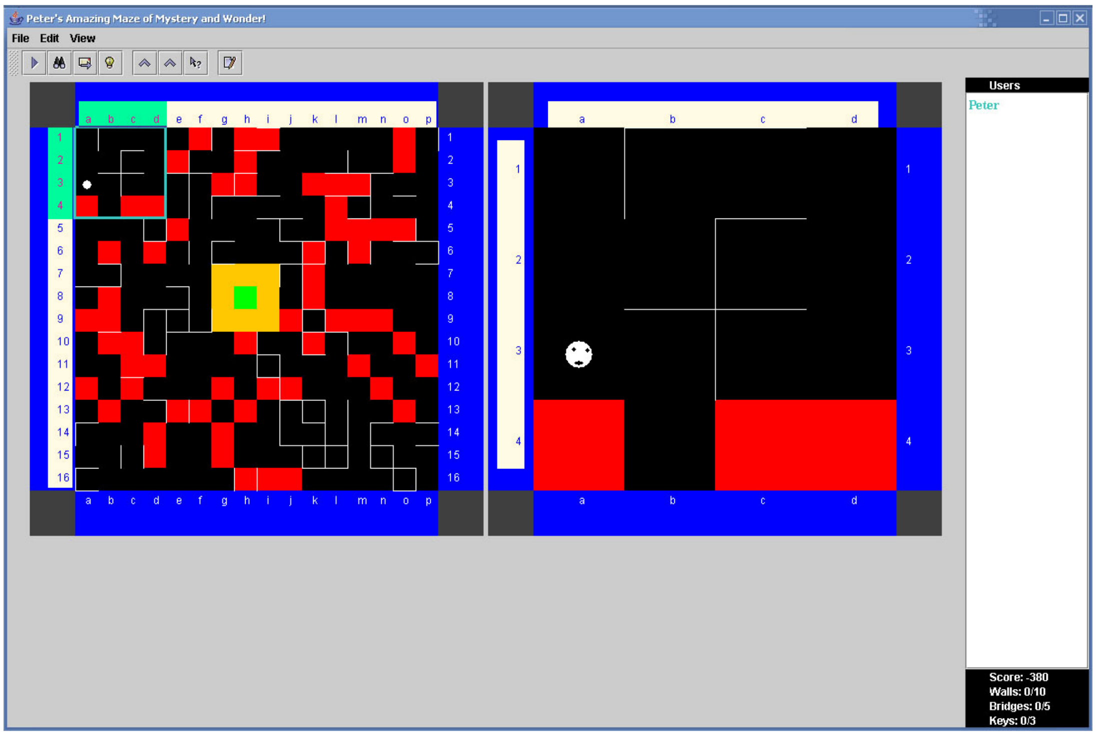


FIGURE 17: ENHANCED RULER VIEW

4.2.3 USING TELEPOINTERS

The next experiment we carried out was the use of telepointers as the main carrier for awareness information. The users would use the telepointers to give the outer users an indication of where they were working in the workspace in a very precise manner. As the user could control whether the telepointer was on or not on the other users screens the telepointer could be used in some more subtle way than was possible with radar.

For example the user could turn off the telepointer when then were working on certain individual tasks and then turn back on the telepointer when they wanted to inform the rest of the group what they were working on.

There were two possible modes I could have implemented the telepointers with. The first way was to use them as pointers where the users could select a location in the workspace and the telepointer would appear there for all the users but would not change until the user selected another location. The second option would have been to implement a continuously updating telepointer that would reflect where the mouse pointer was on the current user's workspace at all times.

The second option would have been very expensive in terms of network usage as we were using RMI over TCP. This would have resulted in a lot of traffic; there would also be a cost in terms of repainting the user interfaces and ordering all these actions correctly. In [30] they designed a system for High Performance Telepointers, HPT. These telepointers were more sophisticated and could perform a number of advanced features like motion prediction. They also recommend that telepointers should be implemented over a more efficient protocol than TCP for example Real Time Protocol – Interactive, RTP/I.

In order to keep the system relatively simple and consistent it was decided to use RMI throughout the system but this would not really support the use of HPT. So the first option of allowing Telepointers that can only really be used as pointing devices was pursued on this occasion.

In the test scenario only the telepointer system was enabled. In this case the users used the telepointers as a sort of pseudo-radar, trying to emulate the performance of the radar system. This can be seen from the image below where by using the telepointers the other users can figure out the range of the user's current view with a degree of accuracy. The only problem is if the users subsequently move location and they do not update their pointer location the awareness information will be out of date.

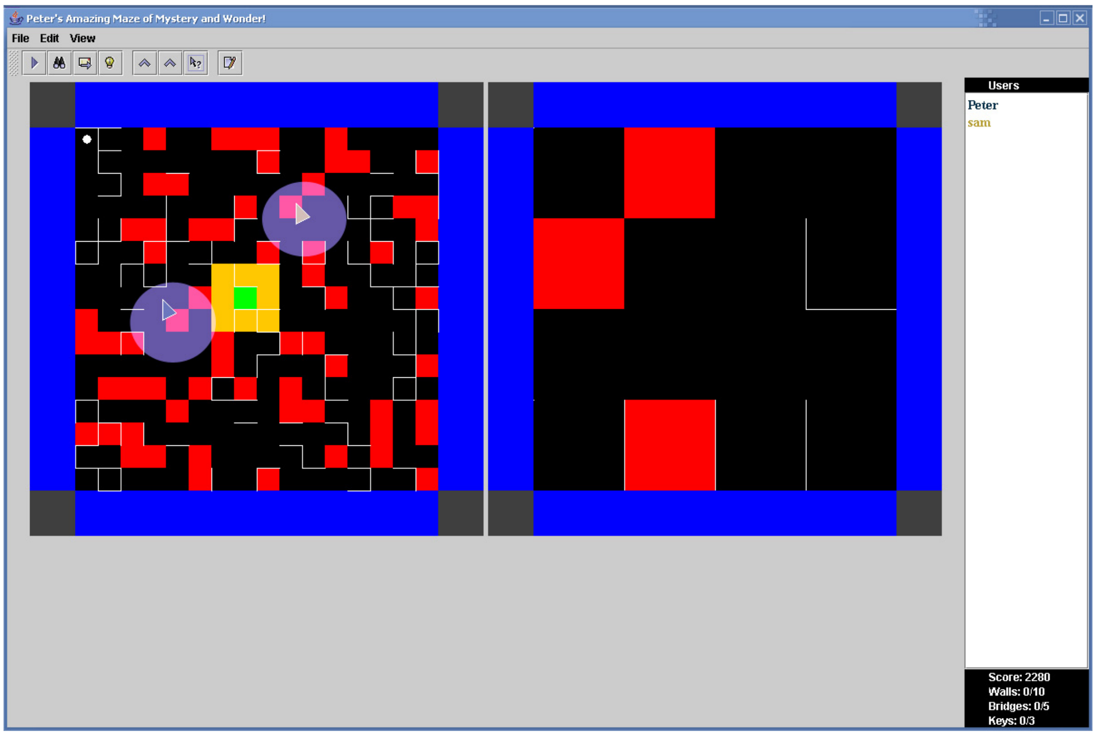


FIGURE 18: ENHANCED TELEPOINTER VIEW

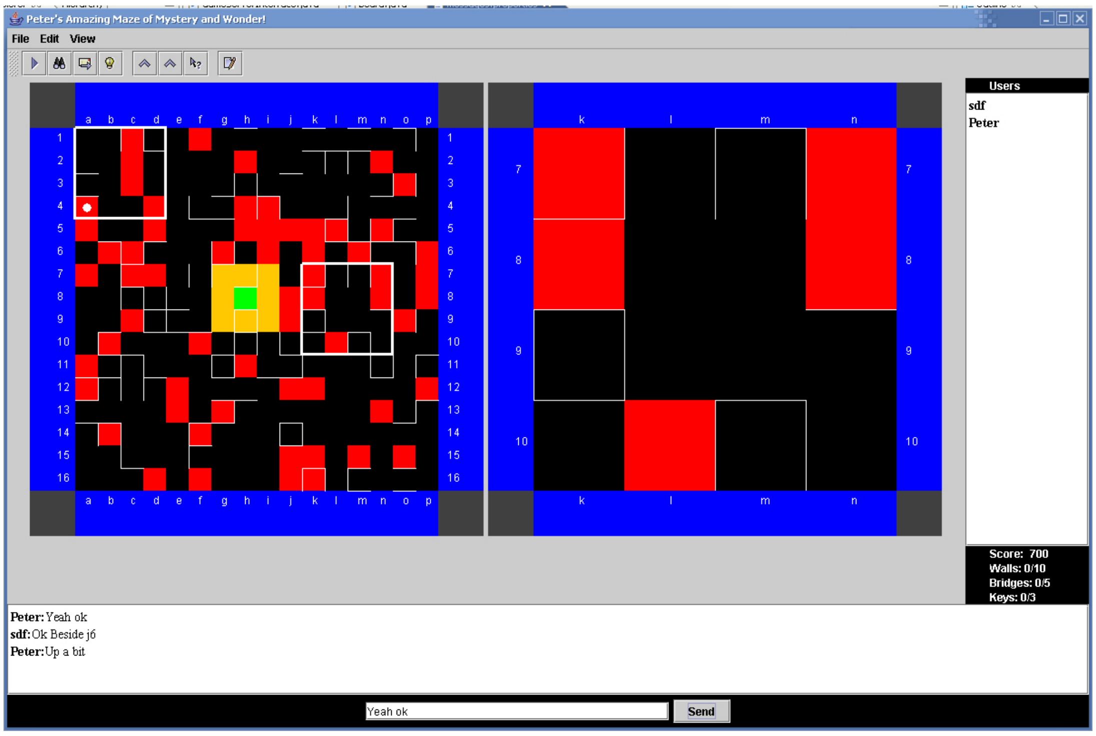
This was the main problem with this particular telepointer implementation. I think the second high performance telepointer implementation would have been the better carrier of awareness information as the users intentions could be gauged more effectively.

As the movement of the pointer would be captured the users could see more effectively the future intentions of the users for example if the user is seen to be moving ahead of the course of the robot that user might be trying to fix the traps ahead of the robot and the observing users could then try and busy himself with another task or develop another route for the robot.

4.2.4 Use of Colour

The use of colour in the system was the most important awareness mechanism, in some respects, as it was the principle carrier of embodiment information in the system. The users could clearly ascertain the identity of the other users in the system and what their actions were. We ran two runs of the program the first run without the use of colour the second with the use of colour.

The players felt more satisfaction with the user interface when users had unique colours assigned to them, for they could tell at a glance who was doing what or who was saying what. Without using the colours on the radar it became very confusing and the radar information was not that much more useful that if there was no radar being used. Users would ask questions like "who is near the top, could you put a bridge over that trap?"



Users
sdf
Peter

Score: 700
Walls: 0/10
Bridges: 0/5
Keys: 0/3

Peter: Yeah ok
sdf: Ok Beside j6
Peter: Up a bit

Yeah ok

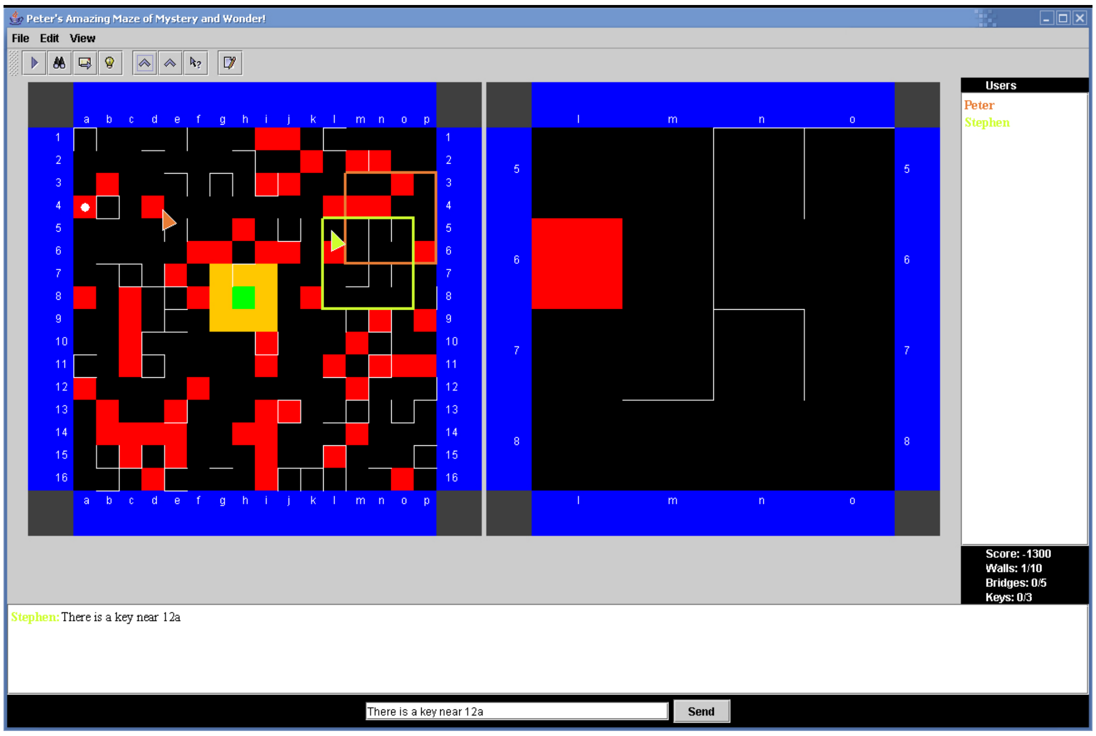
FIGURE 19: NO COLOURS

As can be seen from the above image the radar views of all players are the same colour and the names in the chat window are also the same colour. This can mean that it is harder to tell from a glance what users are doing and it can be confusing to which radar is yours. You also have to look at the shape of the names or read the names to tell who said what. So in conclusion the users seemed to find it a better overall experience to use colour for embodiment information.

4.2.5 ALL AWARENESS MECHANISMS ON

The final test we got a chance to perform was with all the awareness mechanisms turned on. This was a very informative test in this scenario. This scenario gave the users a chance to try different awareness systems and to see which seemed to suit their working style best. It was possible to play the different schemes off against each other, for example did the users use the coordinate system for identifying locations or did they decide to use the telepointers.

It turned out in this application that the users preferred to use the coordinate system than the telepointer in actuality. I think this is because of the way in which the telepointers were implemented, being stationary and not dynamic. It was therefore more convenient to use landmarks in the system as reference points than to use the telepointer in this situation.



Stephen: There is a key near 12a

There is a key near 12a

FIGURE 20: ALL AWARENESS SCHEMES ON

4.3 ALTERNATE TESTING SCENARIOS

Obviously this was only a small trial of the system, and the tests mentioned earlier were more to test the system and its operation rather than to fully evaluate the awareness mechanisms themselves. The system has been tested so that it could be used for the type of experiments and scenarios that would be interesting to carry out. It was a testing platform and the actual testing and evaluation of the awareness mechanisms should be left until later.

Outlined below are some other possible experiments that might be instructive to carry out. Also mentioned is a discussion of the applicability of this system to the more specific tasks of collaborative editing. The analogy drawn between this abstract game system and the more particular task of editing shared documents collaboratively is looked at in more detail also.

4.3.1 Audio Communication

As the system only has a restricted set of possible actions and the workspace is relatively simple to understand it might be useful to use it as a basis for studying audio communication in collaborative tasks as the user interaction should necessarily be a lot simpler than when working on a document or more complex artefact. It should be possible to test the collaborative strategies irrespective of the media domain in some detail with this system.

A comparison of audio and text based chatting could be carried out adequately with this system. As there are a number of different experimental setups that could easily be supported. The first is the use of the chat, text-based, interface only. Another mode would be to allow text based communication and also to use audio communication, this could be that the users are co-located or that they are using some communications mechanisms from outside the system, which might be the best option as it would be a good idea to record the conversations that occur. The text-based system will automatically record the conversations and can be viewed as a standard HTML webpage from the server.

An outline of a proposed testing environment is discussed later and has been adapted from [20]. A setup such as this would be the best configuration for testing adequately the awareness mechanisms. Visual communication could also be easily added to the system.

4.3.2 Awareness of the past

This is awareness of past actions by other users in the system. This would be achieved relatively simply by, for example, colouring the walls in the colour of the user who put the wall down in the system, or colour the bridges added to the system in the colour of the user who added them to the system. This would not give a complete view of the historic actions but it would be useful to see if this information added to the users' ability to work effectively.

Where this information would be useful in the system would be when a user notices a wall in a particular position and wanted to know why it was put there, just by looking at the colour the user would know who to ask about it. This function would be equivalent to the situation where several editors are working on a shared document; perhaps the text could appear in the colour of the person who authored it. Then if there is any discussion of the text it would be apparent who did what.

There are a lot of past awareness information types that might be of interest [5], but the main types that are recorded in the system are the history of actions and artefacts. As the colour mechanism is the only carrier of this awareness of past information there is limited scope to what can be conveyed. The users will be able to tell what was added to the system and who added it to the system. They will not be able to tell when this happened though or if some wall has been edited twice, the first edit is lost. The result of this is that it is not possible to fully know the history of an artefact and if a wall was removed it is obviously not possible to should its absence with colour.

In some systems that hope to capture awareness information about past events a more complete set of attributes would need to be recorded. The key piece of information that would need to be captured would be the time of an event. This could be easily enough be implemented as there is already a Vector Time stamping functionality in the system. Recording the time of the events would open the door to several possibilities.

The first advantage of this would be that not only adding of artefacts to the system could be recorded, but the removal of artefacts could be recorded, this could be displayed to the user by allowing them to move back and forward though the changes to the board over time. This functionality could easily be seen in a document context where all actions are time stamped the document could in effect be rewound to the start. Users arriving late could catch up by seeing the document grow again from the beginning. This might give the user information on why certain actions were taken and who took them.

4.3.3 Testing Environment

In this section we will outline a suitable testing environment that can be used for the evaluation of the awareness mechanisms in an appropriate setting. This setup is based on the setup described in

[20] and should be suitable for testing the working methods of small user groups.

The users should sit at workstations that are setup so they are not able to see each others screens. The users would be able to see and communicate with each other. The conversations could be recorded using a recording device of your choice, whether it involves using a standard recorder to record all noise in the room or separate head sets for each player, allowing the channels to be analyzed separately.

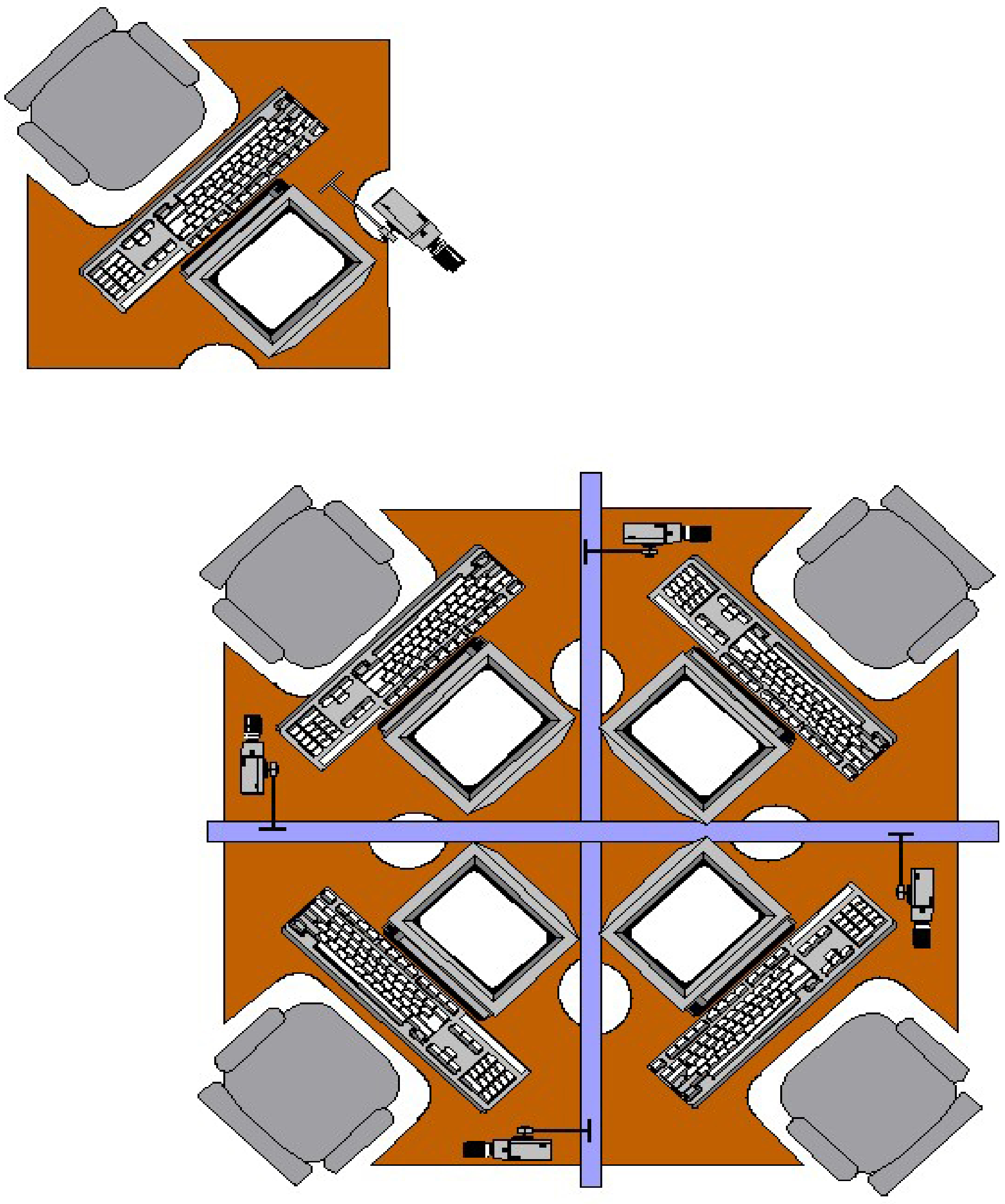


FIGURE 21: TESTING ENVIRONMENT

Actions by the users could be recorded by digital cameras, and possibly a wide shot could take in the entire room for a composite view of the group interaction. The game can be instrumented to record statistics; this could be statically recorded at the end of the session or could also produce live data that can be observed by the experimenter.

5 CONCLUSION

5.1 FUTURE WORK

This section will outline some of the possibilities for future work in this area and how the testing framework could be expanded to include additionally powers. This could include the evaluation of additional awareness mechanisms, improved instrumentation of the system for automatic statistic collection; another possibility would be expanding the workspace to incorporate additional dimensions or features.

Increasing the amount of possible interaction between the users and the methods of interaction so that the system is a thorough analogue for collaborative editing systems should be the priority.

5.1.1 ADDITIONAL AWARENESS

The awareness mechanisms in the system that can be assessed with this framework are embodiment through the use of colour, radar views, telepointers, and communications tools. In a future version it would be interesting to evaluate separately the High Performance Telepointers [30]. This would require switching some actions to RTP/I for the telepointers but would be easily feasible; the current system would still be able to handle all other functions adequately, including discovery etc.

Another awareness scheme that might be interesting to look at would be to try and capture some more of the time dependant awareness information. For example if the system, probably using logical clocks[9], could record this time information, possibly annotated with real time information for reference, a lot of possibilities would open up for a fuller user experience.

Then it would be possible to capture events in the system as time progressed. For example it would allow late comers to see what has happened up to this point in time so far in the system and who did what to what and from this information possibly capture why without having to ask or even know who would be best to ask in this situation. This is past based awareness information, which would be simple enough to capture.

Other awareness information that might be helpful to include would be to vary the embodiment information to the system. In the actual system the User object is there to hold the embodiment information that you want. It would be possible to include additional attributes to carry more of the embodiment information. For example images of the users. If a streaming protocol is used for communication in the future, for example you could use RTP; if you are already using RTP for HTP, then to send streaming video from webcams over the users systems to each client would be simple enough. This could be used displayed in the client together with their names or in the radar view.

Communications mechanisms could be upgraded to include built in audio, as well as the video described above and improved chatting functionality. If it is all in the one application it could then use the same timing facilities as the rest of the system. This would have the effect that users could also see the past awareness information from these communications channels in sync with the rest of the workspace. As this embodiment information is available to the entire system, it is possible to include it almost anywhere it makes sense to in the application.

5.1.2 Instrumentation

This system has good possibilities for gathering statistics about the usage of the system. The best and simplest place to catch this information would be at the server. The server gets a copy of all communication that occurs in the system and could easily compare the state from each of the clients in the system to find useful information.

An example of some useful information to collect would be the percentage of time each use spends looking at the same view port or overlapping view-port of another user, or the proximity of telepointers of two users on the map view.

5.1.3 Additional Dimension

The current workspace in the game is sixteen times bigger that the view port space. This limit to the workspace is relatively arbitrary and could be enlarged. It could be even expanded to include an extra dimension, for example it could be possible to work in a three dimensional workspace.

It would be interesting to discover how the idea of radar could be expanded and represented accurately in this environment. It could take the form of two radar overlays for plan and elevation, but it might be better as a cube object that floats in the 3D map view, using perspective etc. as a means of showing where in the workspace the various users are located. Below is an imagined view of such a radar system.

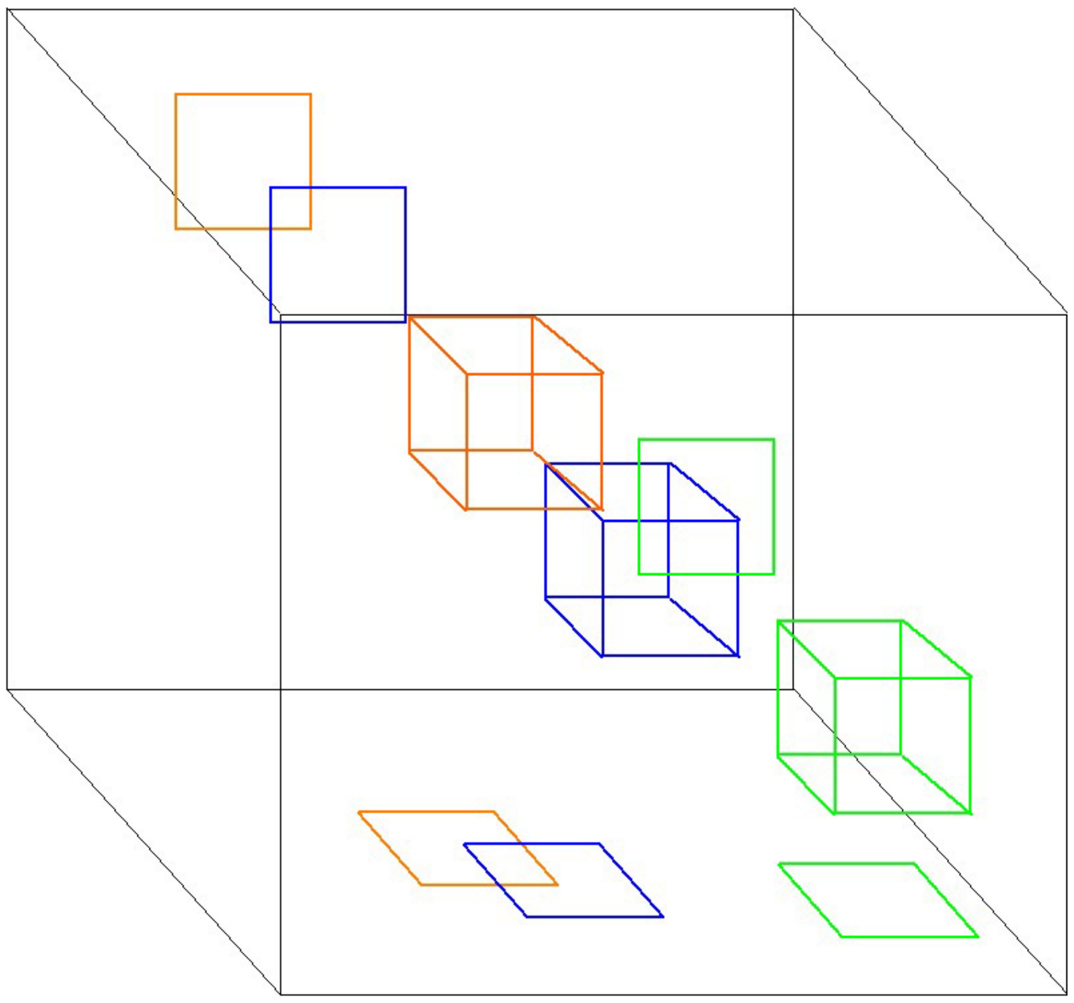


FIGURE 22: 3D RADAR SYSTEM

Another feature that could be looked at is the use of the fisheye view system; this allows areas nearer the users to be shown in more detail on the map view put slowly fade into the distance. A good example of such a system can be seen in [35]. If a larger workspace environment is used in the system then it might be appropriate to use such a system. It could also be applied to the three dimensional system using the same reasoning. In this case you may not see cubes but square pyramids, fading from the base at the front to the apex at the back of the workspace.

5.2 Achievements

In the course of this study we have conducted a small scale pilot trial of the system developed. This uncovered some preliminary results and tested the usability of the system but obviously it would be hoped that this could be expanded to a larger trial in the near future, perhaps allowing for the study of additional awareness mechanisms.

It is our belief that this tool provides a useful analogue between the abstract collaborative game and collaborative editing in general.

It would also be useful to compare this with a more traditional system perhaps a document editor or even a collaborative image creation tool that would incorporate these mechanisms to see how this study would compare and investigate if the conclusions from this abstract study are really applicable to other domains as hypothesised.

5.3 Summary

During this study a framework for testing awareness mechanisms was developed. It was able to test various different awareness schemes including radar system, telepointer, embodiment, and communications. This system was extensible so that it could be augmented so that it can record information about the user's actions in the system.

This study should prove useful for a basis for further research and should be a platform on which it is possible to empirically test awareness mechanisms in a groupware system.

In standard groupware system, particularly collaborative editing systems it is hard to test the awareness mechanisms and other interactions between the users irrespective of the particular task that they have to do. For example, if the writers have worked together before they may have built up a method of working together that works for them, but this is not really good for studying the actual system. They would probably operate in the same manner no matter what tools are provided to them.

Another difficulty when trying to test such systems is that it is dependant on the medium that is being manipulated, for example in a collaborative writing task there are countless ways the document can be assembled, or the workload partitioned between the users.

Writing is not a new task for any user as they have being doing it all their lives, they will then necessarily have built up their own idiosyncratic working methods towards writing documents, some people start by writing the full document in point form and expand on it, others start on the first line and write to the end etc. other people again might prefer to start with the conclusions they hope to arrive at and work back. There could be as many writing styles as users [24].

The same kinds of issues apply to other collaborative editing environments. In an attempt to get away from these personal history concerns and try to discover something more general this framework presents the users with a task that they should be relatively unfamiliar with. The users are presented with the awareness mechanisms to varying degrees so that they can work together, and they will be free to use them as they feel fit, without deference to the past.

6 References

1. Tree-based model algorithm for maintaining consistency in real-time collaborative editing systems claudia ignat, moria norrie Proceedings of the Fourth International Workshop on Collaborative Editing Systems Hosted by the ACM Conference on Computer Supported Cooperative Work New Orleans, Louisiana, November 16, 2002

2. Operation Transformation in real-time group editors: Issues, algorithms, and achievements Chengzheng Sun, Clarence (skip) Ellis Proceedings of the 1998 ACM conference on Computer supported cooperative work Seattle, Washington, United States Pages: 59 - 68 Year of Publication: 1998 ISBN:1-58113-009-0

3. Generalizing Operational transformation to the standard general markup language Aguido Horatio Davis, Chengzheng Sun, Junwei Lu Proceedings of the 2002 ACM conference on Computer supported cooperative work New Orleans, Louisiana, USA SESSION: Group editing algorithms Pages: 58 - 67 Year of Publication: 2002 ISBN:1-58113-560-2

4. Collaborative Editing of XML Documents - An Operational Transformation approach

A.H. Davis, C. Sun, J. Lu Third Annual Collaborative Editing Workshop Preliminary Program ACM Group 2001 Boulder, Colorado USA September 30, 2001

5. A descriptive framework of workspace awareness for Real-Time Groupware Carl Gutwin, Saul Greenberg Computer Supported Cooperative Work archive Volume 11 , Issue 3 2002 Pages: 411 - 446 Year of Publication: 2002 ISSN:0925-9724

6. Evaluating Lock-based protocols for cooperation on XML Documents Sven Helmer, Carl-Christian Kanne, Guido Moerkotte ACM SIGMOD Record archive Volume 33, Issue 1 (March 2004) COLUMN: Articles Pages: 58 - 63 Year of Publication: 2004 ISSN:0163-5808

7. Tree-based concurrency control in distributed groupware Mihail Ionescu, Ivan Marsic Computer Supported Cooperative Work archive Volume 12 , Issue 3 2003 Pages: 329 – 350 Year of Publication: 2003 ISSN:0925-9724

8. Groupware Some issues and experiences C.A. Ellis, S.J. Gibbs, G.L. Rein Communications of the ACM Volume 34, ISSUE 1 (January 1991) Pages: 39 - 58 Year of Publication: 1991 ISSN: 0001-0782

9. Time, Clocks, and the ordering of Events in a distribured system Leslie Lamport Communications of the ACM Volume 21, ISSUE 7 (July 1978) Pages: 558 - 565 Year of Publication: 1978 ISSN: 0001-0782

10. "What Are You Looking At? Newest Findings from an Empirical Study of Group Awareness", Tran, M. H., Raikundalia, G. K. and Yang, Y, Proceedings of the Sixth Asia-Pacific Conference on Computer-Human Interaction APCHI'04, June/July 2004, Rotorua, New Zealand, Computer Human Interaction, Lecture Notes in Computer Science, Vol. 3101, Springer-Verlag, pp 491-500

11. Four principles for groupware design Andy Cockburn, Steve Jones, and JOURNAL: Interacting with Computers, volume: 7, number: 2, pages: 195-210, year: 1995

12. An Investigation of Groupware Support for Collaborative Andy Cockburn, Philip Weir, International Journal of Human-Computer Interaction, 11(3):231--255. Lawrence Erlbaum Associates. 1999.

13. What Is Chat Doing in the Workplace? Mark Handel, James D. Herbsleb, Computer Supported Cooperative Work Proceedings of the 2002 ACM conference on Computer supported cooperative work, New Orleans, Louisiana, USA SESSION:

I M everywhere Pages: 1 - 10 Year of Publication: 2002 ISBN:1-58113-560-2

14. Designing and Deploying an information Awareness Interface, JJ Cadiz, Gina Venolia, Gavin Jancke, Anoop Gupta, Computer Supported Cooperative Work Proceedings of the 2002 ACM conference on Computer supported cooperative work New Orleans, Louisiana, USA SESSION: All ways Pages: 314 – 323 Year of Publication: 2002 ISBN:1-58113-560-2

15. Voice Loops as Coordination Aids in Space Shuttle Mission Control Emily S. Patterson, Jennifer Watts-Perotti1, David D. Woods Computer Supported Cooperative Work Volume 8, Issue 4 (October 1999) Pages: 353 – 371 Year of Publication: 1999 ISSN:0925-9724

16. A Survey of CSCW Systems Tom Rodden Interacting with Computers Vol. 3 No 3 (1991).pp.319-353.

17. A group decision support system for idea generation and issue analysis in organizations planning L.M. Applegate, B.R. Konsynski, and J.F. Nunamaker. Computer Supported Cooperative Work Proceedings of the 1986 ACM conference on Computer-supported cooperative work Austin, Texas SESSION: Session I -

supporting face-to-face groups Pages: 16 - 34 Year of Publication: 1986 ISBN: 1-23-456789-0

18. Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface, Saul Greenberg and David Marwood booktitle "Computer Supported Cooperative Work", pages 207-217, year 1994

19. Crossing from Physical Workplace to Virtual Workspace: be AWARE! Nina Christiansen, Kelly Maglaughlin Proceedings of HCII2003 (June 2003)

20. Effects of Awareness Support on Groupware Usability, Carl Gutwin, Saul Greenberg, Conference on Human Factors in Computing Systems, Proceedings of the SIGCHI conference on Human factors in computing systems Los Angeles, California, United States Pages: 511 - 518 Year of Publication: 1998 ISBN:0-201-30987-4

21. Designing Calm Technology, Mark Weiser and John Seely Brown Xerox PARC, December 21, 1995, Power Grid Journal, v 1.01, http://powergrid.electriciti .com/1.01 (July1996)

22. Viewpoints, Actionpoints and Spatial Frames for Collaborative User Interfaces, Steve Benford, Lennart E.

Fahlén Proceedings of the conference on People and computers IX Glasgow Pages: 409 - 423 Year of Publication: 1994 ISBN:0-521-48557-6

23. Groupware: Shared Thoughts, Shared Media, and Shared Models, James Patrick Williams, LIS 385T Knowledge Management Systems April 22, 2003

24. How People Write Together, Ilona R.Posner and Ronald M.Baecker. Proceedings 25th Hawaii International Conference on System Sciences, Vol. IV, 1992, 127-138.

25. The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility, Mark S. Ackerman Human-Computer Interaction, vol.15, Year 2000, Page 179-203

26. Anchored Conversations: Chatting in the context of a document, Elizabeth F.Chruchill, Jonathan Trevor, Sara Bly, Les Nelson, and DAVOR Cubranic. In CHI 2000 Conference Proceedings, ACM Press, pp. 454-461, 2000, April 1, 2000.

27. Research Issues in the Study of Computer Supported Collaborative Writing, M. Sharples, J.S. Goodlet, E.E. Beck, C.C. Wood, S.M. Easterbrook and L.Plowman, Computer Supported Collaborative Writing. London: Springer-Verlag, pp. 9-28

28. Workspace Awareness Support With Radar Views, Carl Gutwin, Saul Greenberg, and Mark Roseman, Conference on Human Factors in Computing Systems Conference companion on Human factors in computing systems: common ground, Vancouver, British Columbia, Canada, Pages: 210 - 211 Year of Publication: 1996 ISBN:0-89791-832-0

29. A Usability Study of Awareness Widgets in a Shared Workspace Groupware System, Gutwin, C., Roseman, M., and Greenberg, S. Computer Supported Cooperative Work, Proceedings of the 1996 ACM conference on Computer supported cooperative work Boston, Massachusetts, United States, Pages: 258 - 267, Year of Publication: 1996 ISBN:0-89791-765-0

30. High-Performance Telepointers Dyck, J., Gutwin, C., Subramanian, S., and Fedak, C. proceedings of ACM CSCW 2004

31. The Effectiveness of Group Support Systems for Negotiation versus Idea-Generation Tasks: An Experimental Investigation David S. Kerr Uday S. Murthy DSS2004 Conference Proceedings.

32. Effects of Cooperative Interactions on Verbal Communication, Lauren Bricker, Steven Tanimoto, Earl Hunt, and PAPER

submitted to ACM CHI'99, Pittsburgh, PA, and YEAR 1998

33. Exploring Collaborative Navigation: the Effect of Perspectives on Group Performance, Huahai Yang, Gary M. Olson, Collaborative Virtual Environments Proceedings of the 4th international conference on Collaborative virtual environments Bonn, Germany Pages: 135 - 142 Year of Publication: 2002 ISBN:1-58113-489-4

34. Communication in Virtual Environments: Establishing Common Ground for a Collaborative Spatial Task, Ann-Sofie Axelsson, Åsa Abelin, Ralph Schroeder PRESENCE 2003 Programme, Tuesday, October 7

35. Empirical Study on Collaborative Writing: What do co-authors do, use, and like? Sylvie Noël, Jean-Marc Robert, Computer Supported Cooperative Work, Volume 13, ISSUE 1 2004 Pages: 63 - 89, Year of Publication: 2004, ISSN: 0925-9724

36. A Comparison of Traditional and Fisheye Radar View Techniques for Spatial Collaboration, Wendy A. Schafer