Creating a System That Can Draw and Manipulate Shapes from a User's Description in English

Adrian Chandler
Information Systems

2006/2007

Summary

This report describes the design and implementation of my system that attempts to use the natural English language to draw and manipulate shapes for the user, this approach to drawing objects is very different to most drawing programs that predominantly use mouse input to draw what the user wants. The project attempts to use language to produce the final system that will complete basic drawing tasks by processing the natural English language commands entered.

The objectives of the project are:

1. To design and produce a system that understands basic English in order to draw and manipulate a shape.
2. To evaluate how well the completed system works, and look to how it could be improved in the future to include more functions.

Contents

1. Introduction

Currently almost all retail and open source drawing programs for computers use the mouse, and the keyboard for numerical input, as the main user input when using the program to draw anything. Although this may be the most efficient way of producing the desired results for the user, I was interested in whether it would be possible to produce a program that did not use the tradition forms of input but the natural English language instead. With this is mind I set out to produce a system that would allow a user to draw and manipulate shapes using processed English language, given the time restraints I decided to produce only the basics of the system with the minimum amount of functions.

2. Understanding the Problem

2.1 The Problem

Currently the generic drawing programs, either from a computing store or open source, all use the traditional methods, of a mouse and keyboard, to create the desired results on the graphical user interface, they use mouse inputs such as drag and drop as well as precise keyboard entries. This method of drawing through click and drag is not a natural process for humans, we are not always immediately precise when drawing and may take several attempts to create what we want. It is this precision required using traditional drawing methods, either user entry of exact size with the keyboard or drawing with a mouse that I wanted to address. I wanted to create a small program to draw using the natural human language.

The natural human language is not precise and often uses qualitative relations when describing things, in the case of drawing this would be the use of words such as "bigger" or "next to". Current drawing programs require the human to physically move the objects themselves with the mouse or an exact keyboard entry to achieve their goal. This lack of precision though is also a problem when developing a program to understand entered natural language as text, I also have to remove the words that are not needed to perform the operation, essentially all the joining word within the English language.

2.2 Minimum Requirements

The minimum requirements of the project are stated below, the main aim of the project is to produce the system that can use inputted natural English language to draw shapes and this is reflected in the minimum requirements.

1. A system where users can enter English language to:
    a. Create and delete shapes.
    b. Manipulate current shapes size, colour and location.

    Possible Enhancements:

    a. Ability to create 3d objects
2. A user guide for the system built.

2.3 Deliverables

The main deliverable of the project is the working system, but it also needs all the associated documentation and so the deliverables I intend to produce are:

1. A system that can use the English language to create and manipulate shapes.
2. A guide to the system stating its limitations.
3. A report on the system, how well it worked and how it can be extended.

2.4 Qualitative Relations

2.4a What Are Qualitative Relations?

When using qualitative relations, we are tending to refer to things in relationship to others, such as when an object is large then another, there is no precision of how much larger the object is, its all relative. These qualitative relations can be easily seen when looking at coarse topologies when we demonstrate relationships between objects (Egenhofer 1997), looking at these topologies also offers me valuable insight into the qualitative way we think and look at things works.

The process of using qualitative over quantitative to form a program is not a technique often used, and this is mainly because although we may think this way the computers we use do not, as computers use binary and everything is quantitative. This problem though is not one that has not been previously investigated very. However it is said that looking at problems in this different way can establish interesting events and changes (Shama et al 1994), it is this new way of viewing drawing software I want to establish. I have to make software so that a quantitative computer understands these qualitative expressions and ways of entry through parsing; knowing which parts of the data entered it must use and which it must disregard to create the desired results.

2.4b Why Use Qualitative Relations in a Drawing Program?

The natural English language is full of qualitative relations and we use them all the time in our daily lives except when using computers, it is this method of communication that allows us to describe extremely complex situations (Steinhauer 2005). With computer use increasing I wanted to create a program that we could interact with differently, in the way we would with another human. Creation of a program that interpreted human language as text should allow powerful qualitative relations to be used to carry out more complicated processes faster then the traditional drawing methods.

2.4c Problems with Qualitative Relations

Qualitative relations pose several problems when trying to create a program to interpret them successfully, this is due to their ambiguous nature as well as the fact they are regularly open ended (Thorne 2003). This ambiguity is even worse when the program needs a to find a definite command from the entered text to produce any result at all, to achieve this the program must be able to concentrate of the key parts of the relationship described. The use of qualitative relations also means parsing the entered text is harder, as text without relationships in consists usually of only one command on the one object, with the use of qualitative relations the command will need to interact with two objects not just one.

2.5 Natural Language Interpretation

Natural language interpretation by computers is a field of research that has been around for many years and has several uses in the computing world and real life. Although many of these forms of interpretation are of no real interest to me due to the area I am focusing on, there are lessons that can be learn from them. An important thing to take into account is that failure to interpret even the smallest part of the entered natural English language can result in a complete failure to produce anything like the user intended or sometimes anything at all (Division of Informatics Edinburgh University 2003).

The main method of interpreting natural language that I want to use is by using a parser to read in the entered text, cut this up into smaller parts then attempt to interpret these using predefined known phrases and text. This method allows me to reduce the programs interest in unneeded human connecting words as these will not be any use to it. Natural English is also full of grammatical differences; these differences pose huge problems when trying to interpret it with the importance of word order being so great, the process of spotting the grammar is very difficult (Coxhead 2002).

2.6 Why Attempt to Produce a System Based on Natural Entry and Qualitative Relations

Basing a system on natural language input through text as oppose to traditional methods should increase its usability, this is because usually the user is limited to the entry methods allowed. Research shows that people want to use more then this limit allows when "communicating" with the computer. Hauptmann found in his research that allowed subjects to use gestures to help them "communicate" with the computer, that they used more then one finger, or even both hands. This type of interaction is usually limited by the mouse used by computers for manipulation there is only one pointer (Hauptmann 1989). This one pointer is a limitation of the interaction between the user and computer, and thus allowing users to use natural language text to manipulate images should increase the usability of the program.

The use of natural language entry to draw images also has one other major advantage over the traditional methods used; this is the ability for a user to be able to use the system without any experience of the system at all. One of the main factors that is considered when producing a system is the usability of the interface, and large companies try to keep their own products look the same. This makes them familiar to the user even if they have not used it before, the user is also able top predict what will happen when they do things, and as users do not like shocks this is another advantage (Dix et al 2004 page 261). The system I intend to produce should automatically have the familiarity factor as this is a major aim of the solution, the ability to relate directly from real life to the solution. This immediate familiarity will be an advantage over the traditional methods as the user will be able to immediately enter what they want to do, and not have to learn icons used by the traditional drawing program they chose.

2.7 Choosing the Programming Language to Use

Since I would prefer not to have to learn a new language to produce the final piece of software for my project, I decided to look at the two programming languages I know, Python and Java, and then decide which one is most suitable for my requirements.

Python is a powerful interpreted language, meaning it reads the code line by line by the python prompt. Its visual outputs are not very good, and not being object orientated means the only way to create "objects" in python is to use a different set of code, and this is read in by Python but is difficult to save. However, the language itself is simpler to use and being interpreted faults are generally easy to find, the built in functions are vast and also generally easy to work into the program.

Java is an extremely powerful object orientated language; this is to say everything belongs to something (Lyon 2004 page xi). Java itself is more complicated then interpreted languages, such as python, in to the way it works. The standard language involves a lot of symbols, and the commands are generally more obscure and less relative to human words. This said the language is very good for producing visuals and the manipulation of the objects, these can also be easily stored, it can also perform more complicated tasks then some languages such as Python.

Due to the nature of the software I want to produce, with a good clear graphical user interface, and several different objects on that interface being used at the same time, I feel that Java would be best suited to my needs. This is mainly because the type of system I want to produce will involve the production of several different types of onscreen objects at once, all of which will need to be able to be changed independently of each other easily, Java's object orientated approach suits this a lot better then Pythons interpreted one. The main output of the project will also be visual and the need for a full working visual interface is vital, and although Python is capable of producing one, I am more experienced with Java at producing this. I also feel that using Java will allow me to make more use of the built in functions that

Python does not have, and this will allow me to extend my software further then the original design. There are also tools readily available to allow the easier creation of Java, such as Eclipse and Doctor Java; Python does not have the same tools available, and the availability and usefulness of these tools makes Java the better choice.

2.8 Proposed Solution

The proposed solution is a Java program and my justification for this was shown earlier, I propose to create a program that has a small drawing window, an input text field and an error return area as the basic graphical interface, figure 1. The user will be able to enter any single command at once, the program will parse the entered to text, try to locate keywords within the entered text to carry out the process stated on the desired object. If the command is not understood or an unknown object is specified then an error message will be returned.

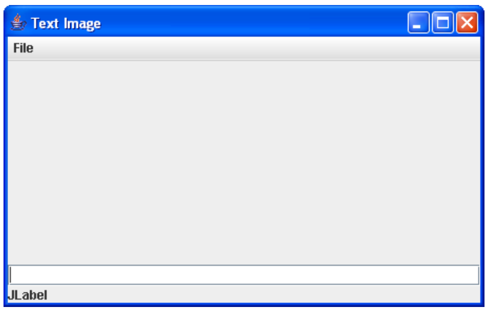


Figure 1
The basic graphical user interface to allow for drawing and text entry.

The program was constructed using three Java files, one for the graphical interface and user entry being read, a second for the object being created and a third for the parsing tool, with the parsing tool was the most important part of the program.

The parser within the Java program, will be based on English language sentence construction, this is to say that the entered commands must contain both a noun, the object being referred to, and a verb, the desired action to be taken. This approach to the parser was for one main reason, this was because allowing more complex entry would make the program simply to difficult to code in the given time period. So the parser is limited to one object and one action per entry; entry of a second action or object will override the first one. The system will also show the user the last statement they entered and an error message if the statement was not understood. This means the user can easily determine if the error was because of a simple human error or the computer not understanding what was entered, this will be based on Norman's Model of Interaction (Dix et al 2004 page 126).

To allow for testing of the program I intend to add several functions to the program to allow for multiple manipulations to be possible to the objects within the shape file. The minimum functions required will being able to move the object about and change both its size and colour. The program will also contain the ability to create two different shapes at first, a circle and a square. I also intend for the proposed solution to be able to turn qualitative statements into drawing commands on the screen; this method would hopefully be faster then the traditional method of click and dragging. It is this advantage when uses qualitative statements such as "next to" that the proposed system will have over traditional methods.

2.9 Difficulties Expected When Creating Proposed Solution

A major factor that I have to take into account when creating the proposed solution is that not all English words have only one defined meaning, as well as different people constructing their sentences differently. The method of parsing, then producing an outcome from the parsed answer is susceptible to errors here, where an unneeded connecting word could be misinterpreted and so the wrong result is produced. Research into using speech for image manipulation back in 1989 showed that of the 223 words used by subjects to manipulate images, only 144 were actually classed as useful (Hauptmann 1989). This difference in words used and useful words means making a program that can differentiate between them is the biggest task.

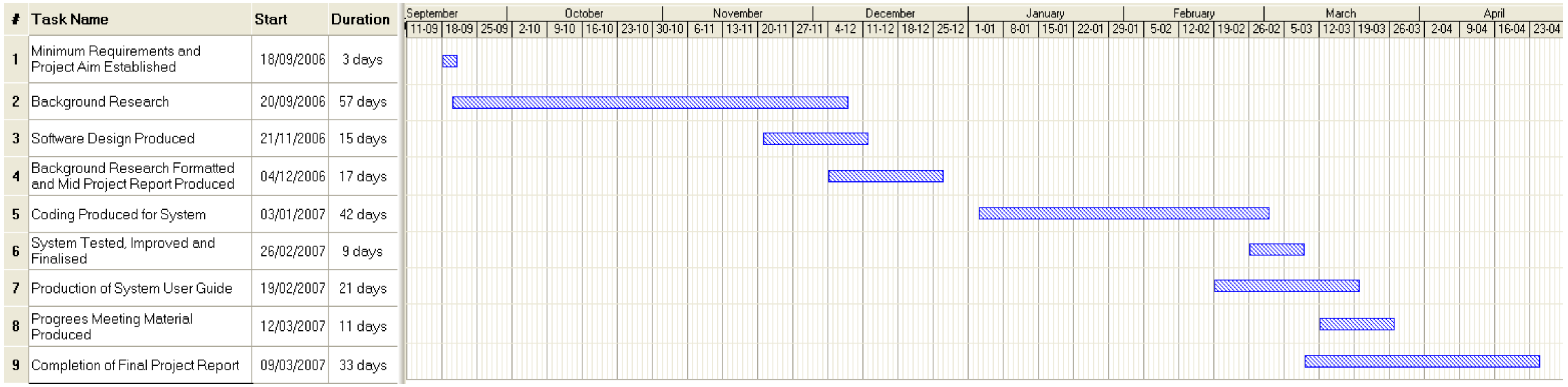
There is also the issue of the user entering the wrong command, or a typing mistake when using the system while learning it, or even performing a repeated task (Dix et al 2004 page 127). The system will have to be strict enough to catch these errors when they are made and return a suitable error message, as avoiding carrying out all unwanted tasks is the ultimate aim.

3. Project Management

3.1 Project Schedule

The initial project schedule was found to be speculative (Appendix D), and the time given to complete each task found to be poorly estimated. However the schedule was mainly changed due to the projects change in main goals, this meant a rework to add in new sections of the system and written work, as well as removing the older redundant sections. As the project was being completed the new schedule, shown below, was kept to with virtually all major deadlines being met on time. The main system was running before initially planned and as a result I was able to move on to other areas of the project, including testing the system more to find ways of enhancing the program, as well as the basic testing it to see if it met its minimum aims. This extra time was used to improve the main problem I found after the system was initiated, this is described later.

3.2 Revised Gantt chart



#	Task Name	Start	Duration
1	Minimum Requirements and Project Aim Established	18/09/2006	3 days
2	Background Research	20/09/2006	57 days
3	Software Design Produced	21/11/2006	15 days
4	Background Research Formatted and Mid Project Report Produced	04/12/2006	17 days
5	Coding Produced for System	03/01/2007	42 days
6	System Tested, Improved and Finalised	26/02/2007	9 days
7	Production of System User Guide	19/02/2007	21 days
8	Progrees Meeting Material Produced	12/03/2007	11 days
9	Completion of Final Project Report	09/03/2007	33 days

3.3 Project Milestones

06/12/2006 – Software Design Completed

21/12/2006 – Mid-Project Report Completed

07/03/2007 – System Completed

24/04/2007 – Final Project Report Completed

4. Delivery of the Solution

The solution proposed was created on schedule, in Java, and met all the minimum requirements, the system however did not carry out any of the enhancements proposed if time allowed. A complete user guide for the system is also available (Appendix B); this guide shows all the operations the system is capable of performing. The system is able to recognise several different methods of English entered into it to complete the tasked asked. There is also a list of the current words the system is able to recognise while trying to perform image manipulation in appendix C.

4.1 Initial Solution Problems

My biggest problem was the initial solution developed was only able to recognise one object and one command per entry. Although this made moving the object, changing its colour and size both possible, there was no way of allowing qualitative relations to be entered into the text box. As the main idea of the drawing program was the ability to manipulate these objects using these relations in statements, I had to revise the program to make it able to understand some basic qualitative relations.

Another problem I encountered was with my initial parsing technique, this was because the system did not use the whole word when trying to find out the command wanted. I had initially done this to help allow for common spelling mistakes, and as a result there was erroneous results when testing some added functionality. I was therefore forced to make the whole words read in, although this may be the better technique to reduce erroneous actions, it does not allow for any mistakes on the behalf of the user. For this reason the error message system was updated to give back clearer messages to make sure the user knows exactly the reason the system did not produce the result they wanted if the entry had been misread. I did not have time to create an undo function, this would have been the best function to have added to solve the problem of misreads easily.

4.2 Enhancements on the Solution

The original solution plan, was intended to only create and manipulate two shapes on the screen, however the final design is able to create multiple shapes if their colours vary, this allows for many more processes to take place. With the addition of multiple numbers of the same shape but different colours in the project, I added a small orange line around the selected object on the screen, this is to help the user see which shape is currently selected and will be manipulated.

If I had more time having produced the system and used it, the functions I felt would be best suited to be added next would be an undo function; this is something I originally overlooked when producing the system and would be something I would definitely include next time. I concentrated mainly on producing the wanted manipulations, as this would add the main usability I wanted to and solve the problem I intended to.

5. Evaluating the Solution

The evaluation of the solution produced is difficult, as the best direct comparison made can be between a fully functional drawing program and the designed solution. Although this comparison will be useful, it will not demonstrate the main aim of the system, which was to allow users to use natural language to manipulate images and draw. To allow any real evaluation of this, with the user guide written I hope to ask users of different skill to use the system and suggest improvements, as well as giving feedback on the design.

5.1 Does it meet the Minimum Requirements?

The produced Java system, with user guide, produced to solve the problem meets all the minimum requirements set, but does not do the enhancements, the minimum requirements are repeated again below.

1. A system where users can enter English language to:
    a. Create and delete shapes.
    b. Manipulate current shapes size, colour and location.
    Possible Enhancements:
    c. Ability to create 3d objects
2. A user guide for the system built.

The created system was able to, create and delete shapes using entered English Language, manipulate the size of the shapes, making them bigger and smaller easily, with an added feature was that once an object was made smaller or bigger when the opposite command is given it changes more slowly. The system can change the colour of a shape and its location, the movement of the shapes location can be either basic, that is up, down, left and right or using added qualitative relations, the words the system can currently understand are in appendix C. I did not add the added feature of the ability to manipulate and create 3d objects, as I felt this would make the system become too complicated and although would be an added feature it was not really the goal of the system being produced.

A user guide was created for the system, to keep the user guide (appendix B) in the same style of the system that is to allow any English to be entered to achieve the users goal, and so the user guide tells the user what the system is currently able to perform, as well as giving an example way. Due to the user guide being vague on purpose, I decided to include a sorted list of the words (appendix C) the system is currently designed to recognise, this can then act as a reference list for the user.

5.2 Was the System Produced on Time?

The system was produced on time that is to say that the minimum requirements were complete. However the minimum requirements of the system only included two types of qualitative relationship commands, the ability to use these commands being a major goal of the system. As a result although the system was complete, ideally I would have given myself more time to allow the adding of more features to the system. These extra features would have allowed for extra comparison to be made between the new system and the traditional system methods of computer user interaction. While producing the system, I was also made aware from my own usage of some features I had not originally planned to put into the system would also benefit the system. Therefore although the system was produced on time, I would have preferred more time to be given to produce the system, and add extra features as I found necessary after some primary testing.

5.3 Does the Solution Solve the Problem Successfully?

The solution meets all minimum requirements, which means it does solve the basic problem of typed English language entry to draw and manipulate shapes on the screen; the solution also has the ability to use qualitative relations. The main question is whether this solution is better then the traditional solutions in certain cases, and finding ways to show it is or is not is difficult. To make my comparisons fair as possible, I will compare very basic traditional drawing program Microsoft Paint, as this program does not have the advance functions as other programs.

5.3a My Software Solution against Microsoft Paint Direct Comparison

To compare my solution against Paint directly is a difficult task due to the difference of the user interface, as well as the large difference in the functionality available on the different solutions. When initially looking at the two solution options I decided to establish some categories that I could compare them, the choice of categories, and the reasoning for them are shown below, with a comparison table shown below:

Category	Reason
Ability to use immediately	This category was vital due to the nature of the solution I created, as this factor was the main focus of the system.
Speed of shape creation (of the right size)	The creation of shapes of the initial wanted size is an important factor when using a drawing program, as this is the basic operation of the program.
Ease of moving shapes	The movement of shapes is a easy way to compare the two, and also a way to compare the ease of use of qualitative relations.
Ease of colour changing	The colour changing of shapes is important for any drawing program.

The table shown below is a direct comparison between the produced solution and Microsoft Paint by me, this table attempts to compare the programs evenly on several categories.

Category	My Software Solution	Microsoft Paint
Ability to use immediately	With the main aim of the system being to allow a user with no knowledge of the system to use it instantly, the solution software excels at this. The software uses basic English language entry to draw, this allows for a direct comparison between the software and real life.	Without any prior knowledge of the software, a user either needs to "experiment" to learn to use the software, and what the icons in the "toolbox" mean. Or they are required to read a guide (Lakewood) or help file to learn to use the software and then carry out the operations they want to.
Speed of shape creation (of the right size)	The version of software I created only performs one operation at a time, and when a shape is created it is a standard side that cannot be specified. This means that once a	Once a user has learnt to use the "toolbox" system used by most traditional methods, then the creation of the desired shape is relatively easy, and the method

	shape is created, the user then has to make it bigger and smaller step by step until they have the size they want.	allows the user to drag the shape size out immediately. Thus this method is faster.
Ease of moving shapes	Moving objects around using the software solution is easy, although basic movement only allows one direction at a time, before a command change is needed to change the direction of the object, and this can be slow. However, the use of qualitative relations here means a user can move two objects instantly relative to each other such as above or below, the ability to use these relationships can greatly increase movement speeds. This method of movement is also more closely related to real life, and again benefits from an ease of use factor.	Paint is poor at shape movement once the original object has been deselected, this is because the shape becomes part of the background image, and as such to move just the shape require it to be cut out then moved. Moving a selected shape is easy as you need to click and drag, but moving an unselected shape is poor.
Ease of colour changing	Changing a shape's colour is easy in the software solution, all the user needs to do is specify the shape whose colour needs changing and then the colour they want it changing to, this all being done in English without any clicking means maybe shapes' colours can be changed quickly without having to locate them on the screen.	The Paint uses a "fill" tool to change the colour of shapes drawn on the page. The user must first choose the "fill" tool, then select the colour from the "pallet" at the bottom of the interface then click on the object they wish to change the colour of. To do more then one shape the user must keep going back to the pallet to choose the colour then locate the shape they want to change the colour of and "fill" it.

Although the comparison only focuses on four factors of the drawing programs, this is mainly being due to the lack of all the functions Paint has compared to my software solution. However even on these few small areas distinct advantages and disadvantages of the solution can be seen.

The solution produced is better then the current drawing solution available in two of the categories clearly, about even in another but competes poorly on the last. The solution performs worse in the basic creation of shapes of the correct size wanted by the user; the main reason for failure here was because the solution cannot create shapes of a user defined size in one command. Whereas Microsoft Paint uses a drag technique to increase the shape size to that desired by the user. Although it is possible to make one shape bigger or smaller then another using the qualitative relationship between the shapes, this means that operations may become a little faster but it is still not ideal. To create the desired shape in Paint can take only two clicks, one to select the correct shape, and another click and hold to drag the shape to the correct size, however my solution to produce a shape of only a minute increase in size requires the user to enter up to eight commands. This difference in the number of commands for such a minor operation after using the solution for a long period of time would result in a huge increase in the amount of time needed to produce drawings required.

The two areas where the solution is clearly superior to Paint, was with the initial ease of use and the shape movement operations. The initial ease of use is a major factor for the solution, as this was one of the ultimate goals of the solution, the ability to use it without any previous knowledge of how it works (Dix et al 2004 page 263-4); only knowing that it uses a natural English text entry method to produce the desired drawings. The ease of use means that users do not have to spend time learning to use the solution, and as such the solution should be quicker to use initially then a program such as Paint. However the time taken here is different for each user, and although I am unable to experiment into long term use, Paint may become more useful then.

The second area where the program is vastly superior to Paint was with the movement of shapes on the screen, this was because of the methods the program used to select each shape, as well as the ability to use qualitative relations to move the objects. The solution keeps each shape as individual shapes, thus meaning they can be moved easily buy referring to them by their shape and colour. Although the movement are limited to four directions, up, down, left and right, it is also possible to move the shapes relative to each other such as above. Both these factors mean that the movement of shapes is quick and easy. Paint creates shapes and these are selected when created meaning movement is as simple as clicking, holding then moving, however once a shape is placed and another create to move the first object Paint forces the user to use the "cut" tool to move it. This means users need to cut the shape out to move it, the only way to do this is by drawing around it, this is time consuming and very prone to mistakes. It is for these reasons that using the solution is superior to Paint when moving objects, saving the user time and requiring less effort.

In the category of changing shape colour, both systems were even with both methods being almost equally efficient, the solution offer allows the user to change shapes colours by simple telling it which shape they want to change the colour of and what colour to. Paint on the other hand uses a "fill" method, with this method the user chooses to fill a shape, the colour they want and then click on the shape. Although the second way might seem slower at first, which colouring multiple shapes the same colour the user only has to click on each shape one by one, whereas the solution I produced requires the user to re-enter a line per shape. It is for these reasons that I do not believe that either method is better then the other here.

The overall outcome of this analysis shows the solution to be better then the readily available Microsoft Paint, however we have to remember that this is a very specific test due to the lack of a solution that currently provided everything Microsoft Paint does.

5.3b My Software Solution against Microsoft Paint Advantages

Although my software solution may not be as quick and useable for every computer user who wants to draw, this does not mean that it is a failure at all, this is because there are many disabled computer users out there. These users may be unable to use there hands to move the mouse or the use keyboard. This is where my solution is a lot better, although the modern voice recognition software may allow for a user to use Microsoft Paint or similar, my solution would benefit users using this software even more. The solution offers a disabled user who currently uses voice recognition software to use the computer to draw using the same software. Although this may only be a small proportion of computer users, giving them the ability to use a computer to perform the same tasks are full able users even at a slightly slower pace is still a big achievement. I was also unable to find any drawing software that could be used easily by a disabled person.

The other advantage my solution has over Microsoft Paint is the lack of precision needed when using it, programs such as Paint require the user to be very exact with how big they want the shape and where they want it from the beginning. Although this method can be advantageous in many situations there are other times where the user may not want to have to be precise, the software solution I produced allows the user to be as precise as they want. The user can move shapes in the exact place they want or they can use statements such as "next to" to move a shape, this allows the user to quickly move shapes about and can be advantageous over programs such as Paint.

5.3c User Feedback

To allow me to get some feedback on the design of the solution and the way the user interacts with it I decided to have a few people test it and give feedback on a short questionnaire. This real user feedback is the invaluable to deciding on the success of the project as well as finding out ways to improve the solution. A sample questionnaire and the filled ones can be found in appendix E. I was only able to have three people test the solution mainly due to time restraints as taking up people's time to complete the questionnaire and then having to analyse their answers is a time consuming process which I initially did not give myself time for.

The user feedback I received was to be expected after testing the solution myself; they found the new method of text entry to draw was a lot more time consuming then the traditional method and as such preferred the traditional method. It was also said that the method entry was extremely strict, and it meant that sometimes commands had to be entered twice to change a small spelling mistake, this also goes against my plan to make the program need less accurate commands to produce what the user wanted as explained earlier. The testing results also show that a user would probably prefer a speech to drawing as they thought it would be a faster and easier method for the user. This is interesting to me as I considered speech for a disabled user and how easy it would be to implement using software already readily available, I did not think of how this would increase the speed for normal users to. The only other real suggestion for an improvement was the ability to use 3d objects, although I did consider this at the start of my design I considered it far to complicated to implement.

The feedback I received on the command sheet (appendix C) and user guide (appendix B) were both positive and I am glad they were useful, the ability to use the command sheet or user guide meant that when one of my testers had an aim in mind they could quickly achieve this. They also both acted as a good reference sheet of the words the solution could understand if they were struggling to complete the task they wanted to because they could not find the right words for the solution to understand. One tester specifically said that without the guide and sheet (as he had both) he felt he might have been searching for the correct words to use for a long period of time This problem should be less of an issue in a much large version of the solution where the solution would recognise more words.

Overall I am happy with the tester feedback even if it was only on a small scale, although they all said the solution was slower then the traditional method and for that reason they did not prefer it, the ones with the documentation provided found it very useful and found using the system easier with them. One tester also mentioned being able to speak to draw as something they would have liked to see, and from my own evaluation I also saw this as a potential development area.

5.4 Areas for Improvements

Having had three people evaluate my solution and having compared it myself against Microsoft Paint, it seems the main area for improvement would be the entry system, not completely redoing it as that would essentially say the project failed and I do not think it has. The entry system needs to be improved in two ways, firstly the language it is able to understand needs to be vastly increased, this is not a major issue as this would increase if the solution was developed further. I would also add an undo function as personally I felt the lack of this function made correcting mistakes a lot more time consuming then necessary. The other improvement would be using some voice recognition software to enter the text for the user, thus meaning the system is a lot quicker and easier to use, this would be the main area for further development of the solution.

7. Bibliography

Coxhead, Peter (2002), *An Introduction to Natural language Processing* http://www.cs.bham.ac.uk/~pxc/nlpa/2002/AI-HO-IntroNLP.html [last accessed 09/03/07]

Division of Informatics Edinburgh University (2003), *Human Communication* http://www.cogsci.ed.ac.uk/~alex/HC1h/flyer/ [last accessed 09/03/07]

Dix et al (2004) Prentice Hall, Human-Computer Interaction Third Edition

Egenhofer, Max (August 1997), *Query Processing in Spatial-Query-by-Sketch* sec. 4.1 *Coarse Topological Relation,* Journal of Visual Languages & Computing Volume 8, Number 4, pp. 403-424(22)

Hauptmann, Alexander (May 1989), *Speech and Gestures for Graphic Image Manipulation* http://portal.acm.org/citation.cfm?id=67496&dl=ACM&coll=GUIDE [last accessed 01/04/07]

Lakewood Public Library (Date Unknown), *Microsoft Paint!* http://www.lkwdpl.org/classes/MSPaint/paint.html [last accessed 01/04/07]

Lyon, Douglas (2004), Prentice Hall, Java for Programmers

Sharma et al (September 1994), *A Qualitative Spatial Reasoner* sec. 1 *Introduction* International Symposium on Spatial Data Handling, Edinburgh, Scotland, pp. 665-681

Steinhauer, H Joe (2005), *Towards a Qualitative Model for Natural Language Communication about Vehicle Traffic* http://www.ida.liu.se/ext/caisor/archive/2005/015/caisor-2005-015.pdf [last accessed 09/03/07]

Thorne, Sally (2003), *Data Analysis in Qualitative Research,* Evidence-Based Nursing 2000; *3*:68-70

8. Appendixes Page

A. Personal Reflections

Having now completed the project I have learnt a lot, the main one of these is just how hard it is to make a computer understand the natural English language and all the complexities this creates. I have also learnt that planning ahead and organising my own time is vital to be successful when producing a software solution. I feel that I have greatly improved my Java programming ability as I never thought I would be able to produce a program like this. I may also like to add that I have learnt it is always good to keep a backup of whatever work you are doing.

Realising the true extent that creating a program that could understand simple natural English language was something I did not think was even possible for me to complete, but when I started it I found myself simplifying the natural language to much and thus the program itself in its first version was unable to understand any entries with two verbs. The complexity of natural language became apparent when I was trying to make the software understand the qualitative relations and I had underestimated the problems I would encounter. This problem leads onto the second thing I learnt, this was to plan and keep to the plan that was in place, often I found myself having only one week to complete something I had given myself three weeks for, and not because I had been to busy but because I had not done enough work in the first two weeks. This way of working meant that I did not have enough time to add extra functions to my software solution that I feel would have made it a lot more successful and generally improved. Although I have been unable to develop the solution as much as I wanted I learnt a lesson from failing to have the time to do this, and a lesson I will take into my career. I would also advise always leave as much time as you can at the end of a project free or not so full, this means when you run over you are not detrimental to other aspects of the work you are completing.

The improvement in Java programming skills is probably the thing that I am happier about, having never been very good at programming and originally being daunted by the prospect of what I had decided to do, I was actually able to produce some software. Although we are taught to use Java within the school I often felt when learning it that I was not understanding what was happening or why I was doing some things, however learning Java must of worked as when I got down to programming I seemed to know a surprisingly high amount of things. That said there were many moments when I was completely stuck and baffled by why the program was doing what I wanted it do and halfway through the original project I actually scraped my method of parsing entered text and restarted it, this took about half the time the original did. To say this problem could have been avoided would be lying as I felt it was part of the learning curve, could it have been handled better? Yes I believe so basically by managing my time more effectively so I had ample time available to recode without to much pressure.

The last lesson I learnt event though I should not have had to was to always keep a backup copy of any work you are completing, computers are not always the most reliable things and break taking all your work with them as mine did. I lost my code when my computer refused to turn on at all and needed a complete reinstall but was saved by the "sent items" folder on the campus webmail, without this I would have been rushed to complete the software solution, this experience told me to always do what you are told and keep back up copies.

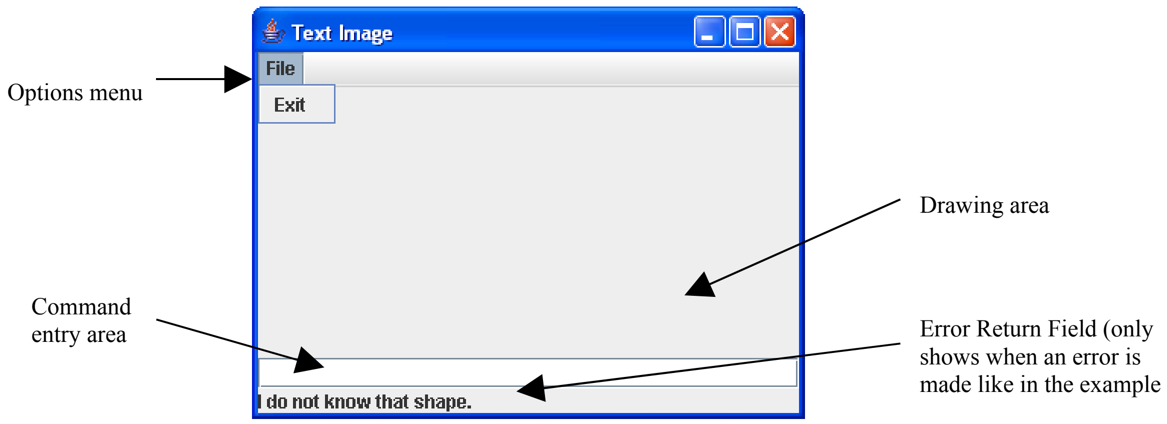
B. User Guide

Contents

1 The User Interface

The basic user interface is made using swing in java, and consists of a draw area; a field to enter the text commands, a return message field also shows when an error message is needed and then a file option to allow the program to be exited without using the "x" in the corner.

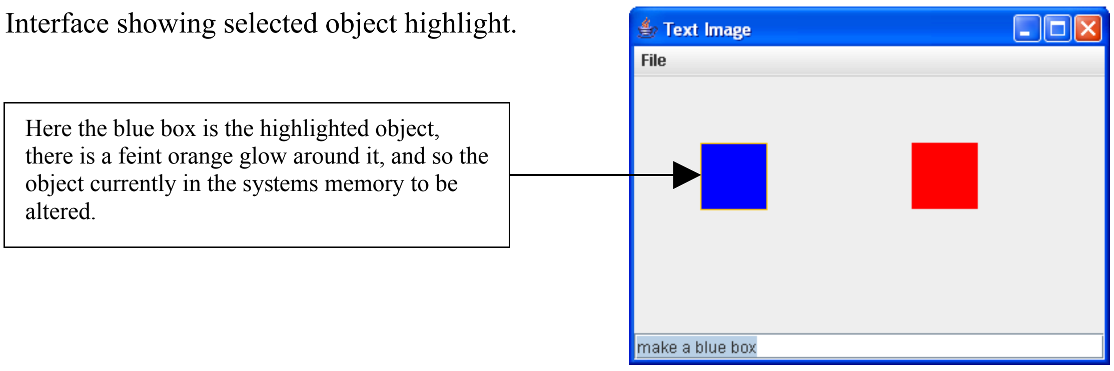
The user interface is shown below (with an error in to show the error field):


Options menu
Text Image
File
Exit
Drawing area
Command entry area
Error Return Field (only shows when an error is made like in the example

I do not know that shape.

When a shape is in the drawing area to allow the user to see which shape is currently selected, there is an orange outline around the currently selected shape, and this shape is the one that will be changed it no specific shape is specified when an operation is entered.

Interface showing selected object highlight.



Here the blue box is the highlighted object, there is a feint orange glow around it, and so the object currently in the systems memory to be altered.

make a blue box

When a command has been entered into the command area, it stays there for the user to see so they can view their previous command, pressing enter will also repeat the command, but whenever new text is typed the old command is overridden.

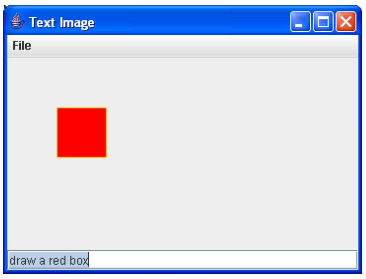
2 Commands

The commands given in this user guide are not the total amount of commands available to manipulate the shapes on the screen and are only a guide of working commands within the program. All commands entered into the command box will be parsed and if a known command is found then it will be carried out, the command box can also only do one command at a time and entering two will result in only the second command being processed. If an unknown command is made, or an unknown or no object is specified then an error will be returned.

3 Creating a shape

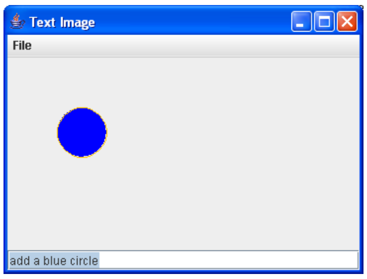
The current version of the program only contains the ability to produce two different shapes; these are circles and boxes (these are squares, except at large sizes). To create a shape the user inputs into the text box a command.

Two Examples



draw a red box

Or

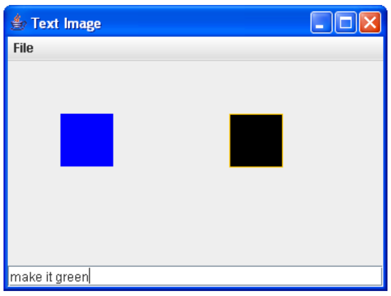


add a blue circle

4 Changing a shapes colour

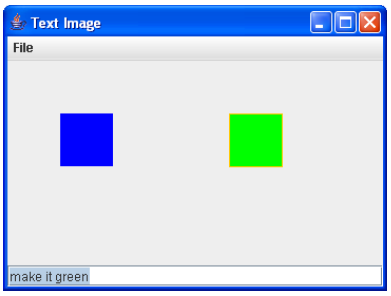
The current system will allow for any shapes colour to be changed, the colours available are red. blue, black, green and yellow. To change a shape's colour there is two methods, the first one is for only when the shape currently selected is the one you want to alter, this can be done using "make it <colour>".

Example,



make it green

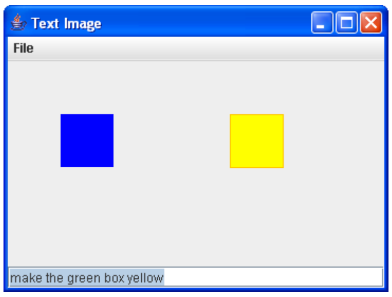
This command changes the selected box (the black one) green, as shown on the left.



make it green

The other way to change an objects colour is to specify the object in the command, this can be done using "make the <object> <colour>" for example, and other commands will work to.

Example



make the green box yellow

5 Moving a shape

Any shape can be moved around the screen in the four basic directions, up, down, left and right. This command can be specific again or just refer to the selected objects, again there are more then one way to do this, "move it down" will move the last selected object down. The objects can also be moved relative to the other objects on the screen, such as underneath.

6 Changing a shape's size

Changing an objects size is an important feature of the system, and for example, once an object is made bigger if the size increase is to large the object will slowly get smaller to the original size again to allow for a wide choice of sizes.

C. Current Action Words Recognised

Movement:
"move"
"put"
"left"
"right"
"down"
"up"

Qualitative Movements:
"next"
"beneath"
"under"
"underneath"
"beneath"
"above"
"top"

Size:
"size"
"bigger"
"smaller"

Size Changes
"increase"
"decrease"

Delete:
"delete"
"remove"

Add:
"add"
"include"
"draw"
"create"

Colour:
"colour"
"red"
"blue"
"green"
"yellow"
"black"
"white”

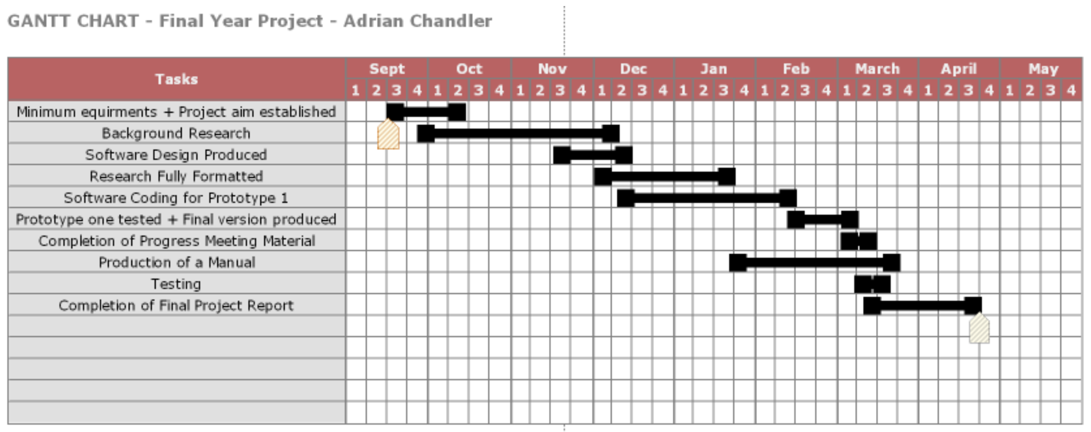
Again:
"more"
"again"

Size of Increases:
"fine"
"coarse"

Last-shape Selection:
"it"
"its"

Shapes:
"box"
"square"
"circle"

D. Original Gantt Chart



KEY

▨ Milestone marker - start

▨ Milestone marker - end

▭ Gantt bar

Key Dates			
15/12/06	The end of the background research, all collabrated.		
25/12/06 -20/01/07	Break due to revisions and examinations		
15/02/07	First Working prototype finsihed		
05/03/07	Software completed		
16/03/07	Progress Meeting		
24/04/07	Final project report completed and handed in		

E Sample and Filled Questionnaires

Having used the system…

1) Do you prefer this method of entry to draw shapes or the traditional mouse and keyboard approach?

2) Even if you do not prefer this method do you like it and why/ why not?

3) What advantages can you see for this method of drawing over the more traditional method?

4) Except the basic functionality you'd expect from a drawing program is their anything else you would want to see included?

5) A. If you had the user guide and command sheet did you find it useful?

   B. If you had just the command sheet did you find it useful or would you have preferred the complete guide?

   C. If you did not have either did you feel you needed them?

6) Any other comments?