G52GRP Interim Group Report

A Multimodal Interface for Free-Hand Graph Drawings

09 December, 2011

Group ID : gp11-exo

Group Members :
- **Tarek Hassoun (txh00u)**
- **Kenneth Jose Bautista (kjb00u)**
- **Tariq Slee-Egeler (txs20u)**
- **Xiao Wang (xxw01u)**
- **Jie Li (jxl11u)**
- **Thomas Alexander Bourne (tab00u)**

Supervisor : Dr. Ender Ozcan (exo)

Project Description:

The project's purpose is to create an easy to use, intelligent interface for creating and modifying graphs, with the added functionality of running various algorithms on the created graphs. The input to the system will be through two different ways. The first is by using a touch screen to draw directed and undirected graphs through the creation of two shapes, circles to represent nodes and lines to represent edges, which could have a direction in directed graphs. The second input method is through the voice recognition sub-system which handles assigning the weights to the graphs' edges.
In the first type of input, using the touch screen to draw the graphs, the user will be able to create the nodes and edges of the graphs by free hand-drawing, which allows the user to input the shapes that create the graphs with greater flexibility and in a larger range of ways. Then the user will add weights to the edges of the graph by talking to the program through the voice recognition sub-system. Finally, the user will be able to apply a number of predefined algorithms, like Prim's algorithm, Dijkstra's algorithm or kruskal's algorithm, on their created graphs. By applying the algorithm the user will trigger two functions in the program. The first is that the algorithm will be graphically simulated on the drawn graph through a color-change of the nodes and edges. The second function, is a new window which will appear giving the user different information about how the graph's data was processed, e.g. the order of nodes explored by the algorithm and the final result of that algorithm. All these features must be implemented in this system, as it is meant to be used in the teaching process, and these features will help make teaching graphs and algorithm more interesting either through the easy, simple and multimodal input or through the colourful and interactive output.

Background Information:

-Existing Products:

Given the fast growing market of tablets and other handheld touchscreen devices, we can say that there already are a lot of existing products that can recognise free-hand drawing and have the ability to do command using voice recognition. However, there are only a few applications that uses these features to be able to simulate different types of algorithms using graphs. One of the few products available was the project that our supervisor Dr. Ender Ozcan did with his colleagues. Their project mainly implements a Unimodal Free-Hand Graph Drawings application that allows the user to draw nodes and edges in a canvas and imitates that of when a person is drawing on a paper using a pen. Having the ink-recognition as the base of the application, they added other features such as 'Area Selection', by drawing a circle around a node, it selects the area including the node which enables the user to move the nodes around the canvas giving them more functionality to save time trying to delete and redraw shapes.

Since our supervisor's project is a unimodal application which combines WIMP-based (Window, Icon, Menu, Pointer) and pen-based (free-hand ink-recognition) features, the scope is smaller compared to that of a multi-modal application in which we are trying

to create for our project. One of the differences in functionality will be that, our multi-modal application will take in values for the edges using voice recognition; compared to that of a unimodal application which can only take in values using a small 'numpad' in which the user can select the numbers for the value they want, or write down the value and let the application recognise the numbers written by the user. Given these difference in functionality, we can say that a multi-modal application gives the user features that will enable them to do tasks quickly compared to that of a unimodal application since dictating a value for an edge can be a lot faster than tapping numbers in a 'numpad' or writing down a number and and getting the wrong input because the recogniser recognised it as something else because of poor handwriting. Although the same can also be said for the voice recognition as the recogniser could have problem recognising different pronunciations of certain values by different users.

-results of technical research into suitable platforms, tools, technologies, algorithms, data structures, etc:
When we first saw the project and discussed it with our supervisor, we were told that we can either implement it using C# or Java. However, only a few in the group could actually use C# and that teaching the other members and actually learning the language could take some time. After having an informal meeting regarding which language to use, we all agreed to use Java as everyone has been using it since our first year at uni. We were also told by our supervisor that there different APIs available over the Internet which we could use for the voice recognition part of the project. After doing some research on which voice API is easier to understand and use, we decided with the one that our supervisor recommended as it seems to be the most suitable one to use, given that we do not have enough experience with implementing APIs such as that of a voice recogniser.

There are several tools and technologies that allows users to draw using the touchscreen functionality like tablets, smart-phones, touchscreen monitors, etc. However, all of these are just simple mouse clicks, mouse press, and dragging the mouse around a canvas to draw something. So, the actual implementation of the project can be done using a normal without the need of any touchscreen devices.

Recognition algorithm:

Pseudo-code for differentiating a circle from a line

```
public double startAndendLength()
{
        double x, y, length;

        length of x line = end point – start point;
        length of y line = end point – start point;
```

```
        length of line = sqrt x² + y²;

        length of line = length of line * 100;
        round of length of line by 2 place;
        length of line = length of line / 100;

        return length or line;
}
public double boundingBox()
{
        double x, y, length, refactored_length;

        length of x line = maximum value of x – minimum value of x;
        length of y line = maximum value of y – minimum value of y;

        length of line = sqrt x² + y²;

        length of line = length of line * 100;
        round of length of line by 2 place;
        length of line = length of line / 100;

        refactored_length = (5.8 * length of line) / 10; // 5.8 is changeable to find the
desired length to differentiate the two.

        return refactored_length;
}

public boolean isItCircle()
{
        boolean circle = true;
        if(startAndendLength() < boundingBox())
        {
                return circle;
        }
        else if(startAndendLength() >= boundingBox())
        {
                circle = false;
        }
        return circ;
}
```

And here is the pseudo code for 2 of the 3 algorithms to be implemented as a sample to show the initial research into what and how the algorithms will be implemented

Prim's:

Prim's Algorithm is a greedy algorithm that is used to find a minimum spanning tree; it takes in a non-empty weighted graph and returns the tree. The edge weights can be negative in this algorithm.

Input: Graph (Nodes, Edges<start node, end node>)

Variables:
Dist[] - distance from the start node to the current node? array of nodes and distances?
Edges[] – array holding all of the edges and their weights?
U<Node, weight + distance> - List / heap of current Nodes, with their weights
F<Node, weight + distance > - List of closed Nodes
Tree[] – data structure holding the information about the minimum-spanning tree

Set all nodes to have a distance of infinity.
For each node n in Graph:
Dist[n]:= Infinity; (Unknown distance function from source to n)
Previous[n]:= Null; (Previous node in optimal path from source)
End for;

Pick a node to be the source (called s).
Since no edge is needed to add s to the minimum spanning tree, its distance from the tree is 0
Dist[s] = 0

While (U still contains some nodes) {
Choose node, n, in U with the lowest weighting + distance
Add n to F

This loop looks through every neighbour of n and checks to see if that neighbour could reach the minimum spanning tree more cheaply through n than by linking through a previous node; i.e. this looks for the node with the smallest weighting to it from the node n not the source node.

For each edge of n, (n1, n2) {(provisionally called n1/n2... could be more or less)
If (length (n1, n2) < dist [n2]) {
Dist [n2] = length (n1, n2)
Edges [n2] = n1
Update U to include the next node
} end if
} end for
} end while

We should now have a structure holding the minimum-spanning tree

Dijkstra's:

This Algorithm finds the shortest path through a graph from a given source node. It is also a greedy algorithm, and so checks all nodes available before it gets to and expands the last node (which this pseudo code assumes is the destination node). This shouldn't be used with negative edge weights and is undirected.

Inputs: Graph (Nodes, Edges<<start node, end node><weight>>) and a source node

Variables:
Dist[] - distance from the start node to the current node? array of nodes and distances?
Edges[] – array holding all of the edges and their weights <<n1,n2><weight>>?
U<Node, total 'distance' > - List / heap of current Nodes, with their weights
F<Node, total 'distance' > - List of closed Nodes
Q – list of all nodes

```
For each node n in Graph:
   Dist[v]:= Infinity; // Unknown distance function from source to v
   Previous[v]:= undefined; // Previous node in optimal path from source
End for;
Dist [source]:= 0; (Distance from source to source)

While Q is not empty:
 n := vertex in Q with smallest distance in dist[] ;
If dist[u] = infinity:
   Break; all remaining vertices are inaccessible from source
End if;
Remove n from Q;
```

This next loop looks at each neighbour of n, called u in this case, and then finds the overall distance from the source node to the new node, if it is less than what was already stored then that is over written and the new path to this node is saved. As the graph is undirected, it is best to know which node you came from, so you don't end up going round in circles.

```
For each neighbour u of n: (Where 'u' has not yet been removed from Q)
   alt: = Dist[u] + Distance between (u, n) ;
If alt < Dist[n]{
  Dist[n]:= alt;
  Previous[n]:= u;
  Reorder n in Q, sort out U and all that...
   End if;
End for;
 End while;
```

Returns Dist[] ;

Requirements Specifications:

we have identified a number of requirements that are in the following two groups:

Non-Functional Requirements:

-To create an easy to use system for creating graphs and applying algorithms to them.
-The system should be multi-modal, having the ability to handle both touch screen input and vocal input.
-The system should be mobile and runnable on a wide range of hardware and software platforms.
-The system should provide the user with useful information about the algorithms, implemented, and how they work.
-The system should be usable by a great range of users, with different skills and abilities.

Functional Requirements:

-The user should have the ability to choose an algorithm out of the initial three algorithms implemented.
-The user should have the choice of the size of the radius of the fixed circle.
-The system should allow for the creation of any number of nodes and edges connecting them to create the needed graph.
-The system should allow for the modification of the created graph through modifying any of its smaller parts, i.e. any of its nodes and edges.
-The system should have a sub-system that handles voice recognition.
-The voice recognition system should allow the user to vocally input the weights of the edges connecting the nodes.
-The user should be able to see an animation representing the application of the chosen algorithm when the algorithm running command is said by the user.
- A new window should open when the algorithm is applied, showing the user useful information about how the algorithm is applied and what are its results.

Initial Design:

The design on the system will be simple and easy to use to keep it consistent with the goal of the system. Initially, when the user starts the program, they will be prompted with the start window, which is a window with multiple choices to a number of parameters that will be passed on to the drawing part of the system, choices like the algorithm which will be ran on the graph that will be created and the size of the circles that represent the nodes if the user choose to single-tap on the drawing area instead of drawing a circle with a continuous tap. After the user chooses these parameters and presses on the submit button to pass them to the latter parts of the system, the start window will close and a new window will open. The new window will contain the drawing area, where the user will create their graph, and the button that allows the user to vocally input the weights of the edges. The drawing canvas is based on JFrame and will allow the user to

draw circles to represent the graph's nodes then to draw lines to connect these nodes which represent the edges of the graph. The user will also have the functionality to hand write the names of the nodes which will be translated into the JLabel object in Java and to draw an arrow on the edges of the graph to make the drawn graph a directed graph. The JFrame that has the drawing canvas will also have a button, created with a JButton object that when pressed the user will have the program listening to them and will be able to say the digits that are the edges' weights. By tapping on an edge or a node the tapped element will be selected and this will give the user the ability to change the weight of the edge, the name of the node or delete the selected element. The voice input button will also allow the user to tell the program to run the selected algorithm on the created algorithm which will activate the animation on the graph. This animation will be the step by step application of the algorithm chosen. Steps like the nodes that are explored then which ones are expanded, the weights of the edges that the algorithm have travelled by and the overall tree that is created by the algorithm through the nodes of the graph. This animation will give the user a better understanding of graphs, algorithms and how they are applied to graphs, and will be exciting animations to draw the attention of users who are an important group of our user base. Accompanying the animations, the user will be able to see a new window opened. This window will contain important information about how the algorithm is being applied to the graph. This information will show how the algorithm is being systematically applied to the graph through showing the list of nodes that are explored, the list of nodes which are expanded, the cost of the path travelled so far in some of the algorithms and the nodes yet to be expanded in the minimum spanning tree algorithms.

Key Implementation Decisions:

The key implementation decision our group made was the choice of programming language. Java is that language in which we are programming the system in. Java allows for easy creation of graphical user interfaces which the project heavily depends on. Java, with its object oriented programming style, allows for the easy development of the free-hand drawing recognition system, as we can recognise the different shapes, in multiple numbers, as different objects and represent these objects on the drawing canvas. The voice recognition sub-system is also easier to create with JAVA, as we are using the MS Speech API. We chose the mentioned API, because our supervisor, Dr. Ender Ozcan, contacted Mr. Metin Sezgin whos is an expert at using the MS Speech API and he was willing to help us with any problems we might have implementing any feature of the API. This project is platform independent, thanks to the JAVA programming language which offers great mobility. So the user can run and use the system on any operating system that has the Java Virtual Machine (JVM). This is clear to us, as the development team, as we have been developing the different parts of the system on different operating systems, ranging from Window 7 to Mac OSX.
The system will also be runnable on any hardware that has the two main functionalities that make this system a multimodal system. A touch screen, to allow the user to free-

draw their graphs and modify these created graphs. And a voice input device, like a microphone to allow the user to give the edges of the created graphs their weights. Most personal computers have an in-built microphone, which means the voice recognition sub-system will have little issue with running on most of the available personal computer in the market. All the development team's computers have this feature. The touch screen part might not has the same availability on the current personal computer's market as a microphone, however, with the rise of the popularity of the tablet personal computers, all of of which are equipped with a multi-touch screen, this system will be available to a wider range of users who own a tablet PC and have the JVM installed on it.

We didn't require a touch screen for the initial development of the system, as a finger touch could be simulated, and will have the same effect, as a mouse-click on a computer not equipped with a touch screen. As for the voice recognition sub-system development, all the members of the development group have a computer equipped with a microphone, which means development didn't require any additional equipment.

The computers used in the development process range from Dell and Toshiba running the Windows 7 operating system, to Macbook Pro and Mackbook Air running Mac OSX Snow Leopard and Mac OSX Lion.

For the development of the Java source code, we chose to use the NetBeans Integrated Development Environment. This IDE gives a lot of flexibility to edit multiple source codes in different files at the same time while also giving the developer the ability to test run the system as we develop it to check for any errors and bugs that may exist in the code, if such errors do exist (and they do), then the developer can run the project in the debugging mode to set up break points and track where the system has a problem or where a small part of the system might go wrong. Using NetBeans also offers an easy way to transfer the developed files from one computer to another, or to take these files and upload them to the version control system for other developers to use. This is done by managing the source code files that exist in the NetBeans project folder in the user's Documents folder. Also, NetBeans, can be downloaded and used on a variety of operating systems, as members of the group have downloaded it on both Microsoft Windows and Mac OSX.

To ensure efficiency in the group work, we are also using a version control system to share the source code of the system between the members of the group. We are using the software Subversion, or SVN, which allows for multiple access to the same source code files so a multiple number of members can work on the development at the same time. Another reason for choosing this specific software, is that the school of computer science has setup a server that could be accessed with it. The server, code.cs.nott.ac.uk is the server we are using to share the files, and this server could be accessed through either logging in to it via any Internet browser, or by using the command-line command svn, which is the Subversion command, to check out files and then commit them back to the repository when finished.

Results of any initial implementation:

The first implementation results we had were the start window and free-hand drawing parts of the project. The start window which is a simple GUI that has one JPanel that is the container for a JComboBox to give the user the different options for choosing the algorithms to be applied to the graph once constructed in the drawing area. The JPanel also has a JTextField which handles the fixed circle radius, the radius of the circle drawn if the user chooses to single-tap on the screen instead of drawing the circle itself. The final element the JPanel has in this part of the system is a submit JButton to confirm the choices of the user and pass these parameters to the latter parts of the system. None of these elements had any use at this point of time as we haven't yet assigned event listeners to them, so they were only for presentation purposes. The start window wasn't connected to the latter parts of the system at this point.

The second part of the initial implementation was the drawing canvas which was created using the JFrame class. At this stage the program could only recognise one shape at a time and if the user tried to draw a new shape the older one would get deleted and the new one would be constructed instead. At first we had a problem distinguishing between a circle and a line and the system drew some parts of the drawn shapes outside the canvas if the user went out of the bounds of the drawing area. The first problem was resolved by the code provided earlier, and later, the system would correctly recognise circles and lines. The second problem was resolved by setting limits to the drawing area and if the user kept drawing outside it then the shape wouldn't be recognised by the system, and the user would be shown an error telling them that they are drawing outside the correct area.

Discussion of problems:

The group have encountered a number of technical and management issues since the start of the project. The largest technical problem we have encountered so far is the implementation of the voice recognition library. None of the members had any prior experience handling voice recognition or anything related to this part of the system. A part of this problem is due to the fact that we had to implement the drawing part of the system before implementing the voice recognition sub-system, and so the member responsible for this part of the system had to wait until the drawing part was implemented. This problem gave us an indication of the failure of the initial work division so we modified the latter to increase the efficiency of work and to involve all the members at an early stage of the project. Other technical problems involved implementing the pattern recognition part of the drawing part of the system, this part didn't differentiate between the circles and straight lines at its initial implementation, however, this problem was later fixed, and now the system can correctly recognise these. Initially we encountered a problem about choosing the interface our system should have (the GUI) and the different windows and canvas the user would see and use when using the system. This was largely due to the fact that we didn't fully understand all the aspects of the system and what will be included in them, however, and after our formal and informal meetings we gained a full understanding of the completed system and we made the decisions on the interface. A problem we have yet to solve is to choose

the data structures into which we will add the created nodes and edges. Currently, the system can only handle one shape at a time, so you can't draw more than one shape on the drawing canvas, this will be solved once we start saving the drawn shapes in data structures, however, we haven't chosen these yet because we are taking into account how these data structures will be used in the two other parts of the system, the voice recognition part and the algorithms part. All these problems have delayed the progress of the project, but they were solved at the end and the overall progress is not disrupted greatly.
As mentioned above we have also encountered some management problems. This was clear when we failed to make some decisions quick enough so that our progress is not delayed. These problems were either due to failure to assign tasks to group members fairly, or due to some group members not attending the formal or informal meetings. The latter problem consisted since the beginning of the project and until now. Some of the members who didn't attend the meetings had a reason for not attending or started attending after other members asked them to do so, while others didn't attend regularly and didn't carry their tasks as the other members did, this was recently brought to the attention of the supervisor and we have yet to see the effect of his involvement. This has greatly affected the progress of our project as these members failed to complete the tasks assigned to them, some of which were relied upon by other tasks carried by working members.

Time Plan:
-Start of the project : 05 October, 2011.
-Start of the programming: 21 November, 2011.
-start of development of the voice recognition sub-system: 1 December, 2011.
-Interim report deadline: 9 December, 2011.
-Start of algorithms' programming: 12 December, 2011.
-End of algorithms, voice recognition sub-system and drawing canvas coding: 1 February 2012.
-Testing: 2 February, 2012 to 16 February, 2012.
-Refactoring: 17 February, 2012 to 30 March, 2012.
-End of project, Final report and software deadline : 30 March, 2012.
-Presenting project at open day: 9 May, 2012.
-Presentation Day: 11 May, 2012.

Reference:

- H. Dibeklioglu, T. M. Sezgin, E. Ozcan. A Recognizer for Free-Hand Graph Drawings.