Master's thesis

Visualisation of and Interaction with Gene Regulatory Networks in Virtual Environments

Menno Visser

Institute for Mathematics and Computing Science
University of Groningen
The Netherlands

Supervisors: Patrick Ogao, Jos Roerdink and Michael Wilkinson

1st June 2005

Contents

List of Figures

Abstract

In the study of biological networks, graph visualisations are often used. Most of these visualisations are 2D, and not much work has been done on doing 3D visualisations of these networks, nor on using virtual environments for these visualisations.

In this thesis we will describe a visualisation technique for gene regulatory networks. The implementation of this will use SARAgene, an application written by SARA and promoted as a framework for biological visualisation applications in virtual environments. This application is relatively new, so we will also make some remarks about its fitness to serve as such a framework. Finally, we propose a new type of multiple selection tool for use in virtual environments. We will show the results from an evaluation and determine the properties that result in the best performance.

There are a number of results in this thesis. First is an extension of the SARAgene application that visualises gene regulatory networks. Also, there is a working version of the multiple selection tool that can be used in any component of SARAgene. Finally, some remarks are made about SARAgene's fitness as a framework.

Chapter 1

Introduction

In biological research, genes are often studied. Genes encode various kinds of information, such as how to construct certain proteins, or they can regulate the activity of genes (including themselves). In this case, they increase or decrease the activity of the regulated gene. If we look at the regulation interactions of all genes in an organism, we can construct a network from this. Such a network is called a *gene regulatory network* (see figure 1.1 for an example of such a network). These networks can be studied to try and understand how the gene interactions result in the more complex processes that occur in an organism. For more information about genes and regulatory networks, see biology textbooks that deal with molecular biology (for example [1][22][5][11]).

When looking at the gene regulatory network, some problems arise in visualising it. The main issue is that such networks can become quite complex, with some nodes containing many links to other nodes. Currently, these networks are mostly visualised using two-dimensional graphics. In this case, it is quite possible for edges to intersect, decreasing the readability of the graph. Also, it is hard to visually link the nodes with other information (for example metabolic pathways) because this will either obscure part of the network, or add more lines that will most likely intersect with network edges.

Virtual reality (VR) environments can solve these problems. In VR, the user is presented with an immersive, interactive three-dimensional environment. Objects can float anywhere in space, and the user can move around in the space and look at objects from a different angle.

1.1 Research objectives

In this research we will aim to do the following:

- Create a virtual reality environment that shows a gene regulatory network.
- Allow users to interact with the network and execute various queries on the data of this network.
- Extract the relevant elements from the DBTBS (Database of transcriptional regulation in *Bacillus subtilis*), a database that concerns itself with the bacteria *Bacillus subtilis* (see section 2.3 for a longer description).

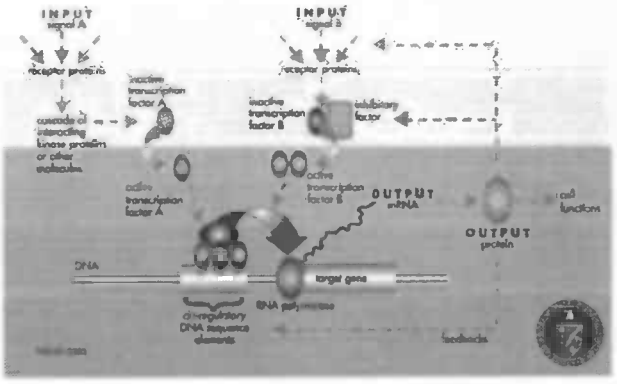


Figure 1.1: Example of a gene regulatory network[1]

- Finally, the environment will be built into SARAgene, an application developed by SARA, and some remarks will be made about its fitness to serve as a general platform for biological visualisations in virtual environments. These remarks can be found in section 6.1.

1.2 Related work

Stolk et al. [20] have created the SARAgene application that can visualise various types of biological data. Among this is a visualisation of a protein-protein interaction network, which is somewhat similar to the gene regulatory network we are working with. They use a self-organising maps algorithm to determine the layout of the network. In [3] some of the advantages of SARAgene in genetics research are described.

Herman et al. [9] give an overview of various graph layout algorithms, clustering methods and navigation techniques relevant to information visualisation for both 2D and 3D environments. This is a useful starting point for finding a good layout method.

Rojdestvenski [17] describes a method for displaying metabolic pathways in 3D. His method includes an algorithm for determining the layout of such pathways, and this layout produces comprehensible views of a given pathway. Our gene regulatory networks share some properties with metabolic pathways, which makes the method adaptable for our networks.

Dickerson et al. [7] use a 3D layout method for metabolic pathways that eliminates edge crossings, and reduces network complexity by having a central *focus node*. Only elements with a short path to the focus node are shown. A similar approach may help in our visualisation.

Nagel et al. [13] identify the various visual properties a data element can have. For each visual property, an indication is given how many data properties can be mapped onto such a visual property, and how large visual differences have to be in order for them to be noticed.

[1] Source: http://cnx.rice.edu/content/m12383/latest/

1.3 Structure of this thesis

In chapter 2, we will discuss the base application we use for this project (SARAgene) and the database we use for our network data (DBTBS). In chapter 3, we will first discuss how we created the virtual environment for the network (section 3.2). After that, details for the types of interaction available to the user will be described (section 3.3). In the next chapter we describe the implementation details of this. After that, in chapter 5 we describe the evaluation performed for the multiple selection tool we designed, and discuss the results of this evaluation. Finally, in chapter 6 we draw some conclusions. Some remarks on SARAgene are made, and suggestions for future work on the application are made.

Chapter 2

Background

2.1 Virtual Reality

Virtual Reality is a relatively new area of research. In VR, the user is presented with a virtual environment with which he can interact. This allows a user to interact with complex data in ways not possible with traditional 2D displays. Using various techniques, the virtual environment is shown in such a way that stereoscopic vision is possible: the user can perceive depth in the scene. Several methods for presenting this environment are available.

One of these methods is a head mounted display (HMD, figure 2.1(a)). In this case, the user dons a helmet which contains two screens, one for each eye, and the helmet hides the outside world. The helmet contains a position tracker which allows the computer to show the virtual world from the correct viewpoint. A problem with this setup is that since the outside world is not visible to the user, care must be taken that the user does not walk into objects. On the other hand, not seeing the outside world also enhances the immersive experience. Another downside is the weight of the helmet, which can be several pounds.

Another method is the immersive projection technology (IPT), such as the CAVE (figure 2.1(b)). This setup consists of a small cubic room (edge length of approximately 2.5m) with either four or six walls. The virtual scene is then projected on these walls. To enable stereoscopic vision, the user dons shutter glasses, which rapidly alternate between blocking light through the left and right glass. When this alternation is synchronised with projecting the left and right image on the walls, stereoscopic vision is possible. As with the HMD, a position tracker is attached to the glasses to ensure the proper perspective is used. IPT has a few advantages over the HMD. First, there is no need to wear large equipment, the glasses are small and light. Second, since the environment is projected on a set of walls, the field of view is larger. Another advantage is that multiple users can wear shutter glasses and see the environment. Although these other users will have a somewhat distorted view, every user can see the others and interact with them.

In VR, various types of input devices are available. A wand device is essentially a 3D mouse. Its position and rotation are continually tracked allowing the user to for example point at items in the virtual world, and several buttons

(a) A head mounted display[1]

(b) A user in a CAVE setup[1]

Figure 2.1: Different setups for displaying virtual environment

allow interaction.

A PDA (Personal Digital Assistant) is another option. In this case, a PDA is used with a wireless connection and a program written for the VR application is loaded. It is then possible to give commands to the VR application via the PDA.

Also, gloves are available. Like the wand, the position and orientation are tracked, but interaction is done by finger gestures.

2.2 SARASIM and SARAgene

In this project, all implementation work is done on top of a software program called SARAgene. SARAgene is an application developed by SARA Academic Computing Centre, and is a program for genomics exploration in virtual reality. One of the design criteria of SARAgene is generic applicability, so that it can be used by academic third parties and can be extended - effectively being a framework. SARAgene was chosen as a basis for this project to test whether it indeed provided this generic applicability. In the rest of this section, a more in-depth description of SARAgene will be given, as well as a description of the framework on which SARAgene is built, called SARASIM.

2.2.1 SARASIM

SARASIM is a framework for Virtual Reality applications[19]. It was originally intended to provide a set of reusable components that would ease the development of all future VR projects at SARA. The programming language chosen for SARASIM is Python[16]. Since Python is object-oriented, this makes it easy to write a modular interface, and the style in which Python works allows for rapid prototyping, another useful feature.

In fig. 2.2 the architecture of SARASIM is shown. At the basis of SARASIM are two libraries. CAVELib[6] by VRCO provides the basic functionality to

[1]Source: http://archive.ncsa.uiuc.edu/Cyberia/VETopLevels/VR.Systems.html

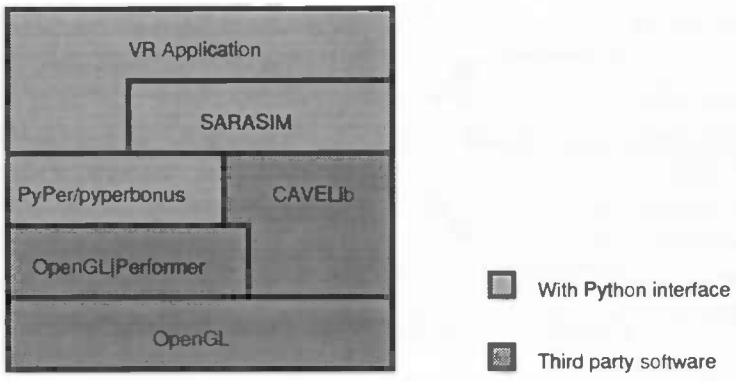


Figure 2.2: SARASIM architecture

make applications for VR environments, such as projection calculations for the left and right eye and the handling of multiple projection screens. OpenGL|Performer[15] from SGI is a scene graph library and also provides math functions.

Because OpenGL|Performer does not come with a Python interface, SARA wrote one. This wrapper is called PyPer. Also, an additional part was written, called *pyperbonus*. This addition provides various features not present in Performer, yet these are useful to essential for 3D applications. Some of these additions include extra geometry objects, a key frame animation system, and an event model.

The event model allows a programmer to link an event handler to any node in the scene graph. Various events are available, such as the node being clicked on, or a periodic clock tick. What makes the event system interesting, is that the events can be shared. What this means is that a SARASIM application can be run on two different computers, only joined by a network connection of some kind, and then events generated by one application will be sent to the other application. This allows multiple users to use a SARASIM application simultaneously at different locations. This functionality was successfully tested at the SC2003 event[18].

SARASIM mainly builds further upon PyPer, adding several modules which provide various functionality. The modules that are interesting in the scope of this thesis are the navigation and menu modules.

The navigation in SARASIM is heavily abstracted, and separated into two objects: an input device, and a navigation model. The application talks to the navigation model to determine the position of the user within the scene, and to this end, the navigational model talks to the input device. Setting up the navigational model like this makes it possible to change navigation model or input device without any hassle.

The menu module provides an immersive graphical user interface. Through an easy interface, the application programmer can quickly set up a hierarchical menu system. This menu is then shown on one of the CAVE walls so it is always easily accessible by the user.

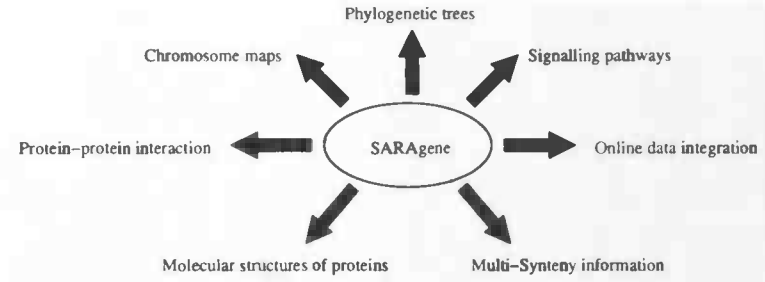


Figure 2.3: The SARAgene components

2.2.2 SARAgene

As noted, SARAgene is a program for genomics exploration and is built on top of SARASIM. It was written to ease the mining of information from genetics databases[20], and incorporates the information from several databases. This information can then be presented using various visualisation methods.

SARAgene is composed of various components. These components can be seen in figure 2.3. We will now give a description of each of these components.

When using the chromosome maps, the user is presented with a two-dimensional panel on which the various chromosomes of a species are shown. It is then possible to make a selection on a part of a chromosome to select the genes within the given range. Using multiple chromosome maps of different species in this way will quickly show homologous genes, meaning that these genes share ancestry. These maps can also display multi-synteny information. This shows if a gene from one species is also present in another species.

Signalling pathways are series of chemical reactions occurring within a cell. These pathways are, like chromosome maps, displayed on a two-dimensional panel. On this panel the pathway network is displayed. This network consists of gene products (proteins, RNA), other molecules, references to other pathway maps, various interactions and relations between components. Some of the proteins will be shown in a green box instead of a white one, indicating that these proteins can be clicked on. When this is done, blue lines from the selected protein will go to the position of that same protein in other visualisations.

When looking at a signalling pathway, it is also possible to click on a protein with a different button to show the molecular structure of that protein. This information is queried from an on-line database, and displayed in the centre of the virtual environment.

The protein-protein interaction component will often be referred to as the MINT map in this document. This is because the current information source for this component is the MINT (Molecular INTeractions) database. This component displays a large network of protein names connected by lines indicating there is an interaction between the proteins so linked.

SARAgene can gather its information from a number of on-line databases. Currently these databases include KEGG (Kyoto Encyclopedia of Genes and Genomes), which provides signalling pathways, PDB (Protein Data Bank), which is a database containing 3D molecular structures of proteins, and En-

sembl, which contains information on chromosomes, such as the genes located on a chromosome and synteny information.

From a programmers' perspective, there are two main additions of SARAgene to the SARASIM and PyPer libraries. The first is a base class for visual objects, which provides functionality that allows the user to grab such an object and place it somewhere else in the virtual space. The second addition is a *connection object*, which facilitates communication between various visual objects and allows them to know the position of certain elements within them. With this connection object, it is possible to show links between different visual objects that have an element in common.

Recently, we have performed a usability case study of SARAgene [14]. In particular, this study evaluated the effectiveness of the MINT map component of SARAgene. From this study, some points for improvement were determined, and these points were taken into account while constructing the additions described in this report.

2.3 DBTBS

The DBTBS[12] concerns itself with a bacterial species called *Bacillus subtilis*. This species is one of the best studied ones in literature, and its complete genome has been mapped. The aim is to determine the complete gene regulatory network of this bacteria, by combining the results of various experiments into one database.

As can be gathered from the name, the database essentially contains a list of transcription-related elements. Promoters that have been characterised in the various experiments used to build the database are listed. For these promoters, transcription factors and sigma factors are then listed. Promoters are the DNA sequences that enable transcribing a gene. Transcription factors are the proteins that bind DNA to a promoter, regulating transcription, and sigma factors ease the binding to promoters.

Currently, the database contains information of 114 transcription factors, and 633 promoters of 525 genes.

In this project we will not use all information present in the DBTBS. Only the information needed to construct the regulatory network is used, which means that information such as binding sites is not used. These sites indicate which DNA sequence the protein binds to, and as such are not relevant to determining the network and displaying it. In future work on the project, we may incorporate them and visualise them.

Chapter 3

Design

3.1 Introduction

The addition of the gene regulatory networks view in SARAgene can be roughly divided into two parts: the actual view that the user will see, and various methods that the user can use to interact with the network. In this chapter we will describe the details of the design process for these parts.

3.2 Network visualisation

One of the main issues in this project was how to visualise the network represented by the DBTBS. The visualisation should help a user of the program to understand the structure of the network. Also, since the human visual system can quickly gather information from an image, we can show certain important properties of the network directly in the visualisation to further increase the effectiveness.

In this section the details of the visualisation of the DBTBS network will be explained.

3.2.1 Layout

If we look at the regulatory network, we see that this can be represented as a directed graph, which can have loops and multiple edges. The main problem for the layout is determining the positions of the nodes in order to get a layout that is easy to comprehend for a user. When determining the layout, we do have a few advantages from using VR. As Herman [9] et al. stated:

- A 3D layout literally gives more "space", making it easier to display large structures.
- A user can navigate to find a view without occlusions.

Also, in 3D the problem of edge intersections is virtually nonexistent.

We will describe two layout methods we examined in this project, of which we chose to use the second one as the actual layout method.

Self-Organising Maps

To determine a good layout, we initially looked at an existing component in SARAgene, the MINT map. Since this is also a network of relations, it might have a useful method. What this component does is use an algorithm called Self-Organising Feature Maps [10] (SOFM) on its data to map each node to a grid point. The SOFM algorithm is a neural network algorithm that groups input samples together based on certain features of the input sample. Input samples that closely resemble each other will end up being close in the output grid, and samples that are very different will end up far away from each other. Essentially, the SOFM algorithm can be used to map any input dimension to any other output dimension. It seemed like a good algorithm to go with, as it would group together genes that are similar to each other.

Unfortunately, there were a few problems with this method. First, the feature data was incomplete: for some genes, not all features were known, and for some genes no features were known at all. This blocked the SOFM from computing positions for all nodes. Also, the output of the algorithm are grid positions. This gives similar genes exactly the same position, but does not give information on how to layout these genes locally, and also gives a suboptimal indication on how similar the genes are to genes at other grid points - the difference is discrete in grid units. Finally, because the algorithm can only handle a fixed number of features for a given input, it's not possible to take into account the edges between the genes. This resulted in a layout in which edges were all over the virtual space without much coherence, reducing the comprehensibility of the graph.

Force-directed layout

The next method that was investigated was a force-directed layout [2]. In another project at our institution a 2D application is being developed to mine data from the DBTBS, and in this application the force-directed layout gave good results in finding a layout for the DBTBS in 2D.

Force-directed methods are often used in graph layout algorithms because they are easy to implement and tend to produce aesthetically pleasing layouts. The graph layout produced with this method gave a clear overview of the network, and this layout method was adopted in favour of the SOFM algorithm.

3.2.2 Element visualisation

When displaying a directed graph, both the nodes and the edges will need to be assigned some kind of graphical element in order to show them to the user. When determining these elements, it is also important to take into account some of the properties of the network, and some of the information that should be easily visible in the visualisation.

Looking further at the directed graph of the network, we see that this graph may have loops and multiple edges. Currently, it is assumed that it can't have more than one loop per node (i.e., a gene can only regulate itself in one way), or more than two edges for a multiple edge (one gene can only regulate another

gene in one way, and the other gene regulating the first gene results in a multiple edge).

Line visualisation

In its basics the visual elements for the edges are rather simple: lines going from one node to another with some kind of direction indication. But straight lines are not a good choice for node-to-node links, since this could overlap the two lines in a multiple edge, making it impossible to distinguish between the two lines. If the lines have a different regulation type, this cannot be seen in this case. Also, straight lines do not work for loops.

The scheme used to show the edges is as follows. If we have a single edge between two nodes, we can simply use a straight line. When we have a multiple edge, we use a Bezier spline. For a loop, a spline is also used.

To indicate the direction of the edge, a small cone is added at the end of the line. Finally, the line thickness is set to such a value that the lines are easy to see.

Looking at the DBTBS data, we see that there is a rather important property is present for each edge: the type of regulation between genes. There are three main values for this property: positive regulation, negative regulation, or a yet undetermined type of regulation. We have chosen to clearly visualise this property by colouring the edges. Positive regulations are coloured green, negative regulations are red, and undetermined regulations are assigned a gray colour. There are a very small number of edges not in these three categories (these have different types of regulations based on some conditions), these are currently coloured as if they were undetermined.

Node visualisation

For the node representation, we need a fairly small element in order not to hide too much of the network behind it. Also, some node-specific information can be shown in this representation, and it should identify each node. Since the name of the node is such a unique identification, this became our node representation. Around the location of the node in the virtual space, we place a rectangle on which the name of the node is written.

3.2.3 Visualisation of clusters

Using various techniques, it is possible to cluster the genes of the DBTBS together. It is useful if the network visualisation can be used to display the results of these clusterings. It should be noted that the clusters themselves do not have relevant names or similar identification and as such this does not need to be displayed. The clustering of the nodes itself was not computed by us, but given to us by our biology department.

We will use two methods that are used together to show clusterings in the nodes. The first method is by using visual properties. In [13], the various different visual properties a static object can have are listed. Some of these

properties are so-called *pop-up cues*, which means that these properties can be used to preattentively identify groups of objects. These properties are:

- **Position** A strong pop-up cue. Position can be used to map three continuous variables.
- **Pose** Pose, or spatial orientation is experienced relative to the user's interpretation of horizontal and vertical in the virtual space. Theoretically it can be used to map two continuous variables. With sufficient difference between poses, it can be a pop-up cue.
- **Size** Larger objects tend to stand out from a group of smaller objects and as such size can be used to indicate a variable. But, since in virtual environments size is also used in depth perception, it is best not to use it to avoid confusion.
- **Shape** Shape can be an effective pop-up cue, although care must be taken when choosing what shapes to use for different values. Some shapes can be distinguished easily, while other combinations do not allow for preattentively discerning different values.

  It is often thought that symmetric shapes are processed more efficiently than non-symmetric shapes, so it is preferred to use symmetric ones.
- **Colour** Colour covers both hue and saturation, and can act as a particularly strong pop-up cue. It is advised not to use colour for displaying continuous variables, since the human visual system is limited in accurately distinguishing between hue, saturation and brightness.
- **Texture** This can be defined as granularity, orientation and pattern, amongst others. Texture aids in determining pose and shape, and can theoretically be used to map one or more continuous variables.

Since preattentive processing helps quick comprehension of visual data, it is preferred to use a preattentive property. Since position is already used in the layout of the network, this property cannot be used. Based on the results of [8], we chose to use colour as the property to indicate various clusters.

Since the current implementation of the text element in SARASIM there is a limitation in what colours can be used as text foreground and background. Because of this, we can't use the background of the text as cluster indication. Instead, we added a small cube above each gene name that takes on the relevant colour. These cubes are hidden while no clustering method is selected, since in this case the colour would be the same for all cubes and as such it would not add any value, but possibly even be a distraction.

With the default layout, nodes that are in the same cluster are not likely to be near each other. This does not help in identifying genes that are related to each other (according to a given clustering). Thanks to the layout algorithm we use, we can use the cluster information to group nodes in one cluster together. Together with the coloured cubes, it is easy to distinguish between the various clusters. The only downside is that in this process, the actual interactions

between genes have to be given a lower priority, and the resulting layout may have many very long interaction lines (as opposed to the unclustered layout which has only a few long lines).

3.3 Interaction

In order to easily study a network, it is not enough to merely display it. The user needs to be able to interact with the network in some way in order to help gain understanding from it or show more information about certain nodes that is normally hidden because it would clutter the view space. We will describe the various methods we designed that a user can use to interact with the network.

3.3.1 Interaction methods

First, it should be noted that SARAgene uses a cursor, displayed as a ray of approximately 1.22m long. When using the wand, the position and location of this ray are linked to the wand, and the user can use this to intersect the ray with various pieces of geometry. Most interaction is done by intersecting some piece of geometry, and then pressing a wand button.

Highlighting

To aid the user in understanding the structure in the denser parts of the network, we have added a highlighting mode. Whenever the user intersects a gene, all outgoing edges are highlighted by changing the line colour to white and using a larger line width. This enables a user to quickly determine what genes are influenced by a certain gene. Incoming edges are currently not highlighted because our network data structure does not enable quickly finding these edges.

Selection

Also, the user is able to select nodes. By intersecting a node and pressing the left wand button, the node will be marked as selected and this is reflected by the node's colours changing from blue text on a yellow background to blue text on a red background. This indicates a selected node. It is also possible to do multiple selection, this is explained in more detail in section 3.3.2.

Filter mode

When a user has some nodes selected, it is possible to use this selection to enable a filter mode. In this mode, all nodes that are not selected are hidden, as well as the lines emerging from those nodes, and any cluster-indicating geometry. This helps users to focus on a small section of the network and is especially useful in dense parts of the network where the many nodes and lines that are present may block having a clear view of other nodes. Of course there is an option to disengage the filtering mode and show the entire network again.

When having filtered out many nodes, a user may find that an interesting chain of gene interactions uses a node that is currently hidden. In order to avoid

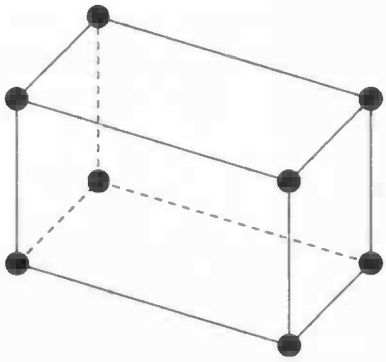


Figure 3.1: The multiple selection box

having to show all nodes, find the larger selection, and then filter the network again, an option has been added that effectively combines these actions into one. The user can select one or more nodes that point to currently hidden nodes, and then choose to unhide all nodes that are influenced by these selected nodes, thus slightly expanding the set of visible nodes and edges. For convenience, there is also an option to unhide the nodes influenced by the entire set of currently visible nodes.

Searching

If a user needs to find a node because for example other research has shown a node to be a promising starting point, there needs to be a search facility. When selecting the search option, the user is presented with an input board. On this board, the user can intersect and then click on letters to enter a part of the name of the node that needs to be located. In a list below the board the subset of nodes which have this string in their name is shown, and the user can click on a node in this list. When this is done, the view moves to put the chosen node in an easy to locate point in the CAVE: centred relatively to the left and right walls, a bit towards the back wall, and slightly below eye height. The current selection is also changed to contain solely the node, which makes it stand out from any surrounding nodes. With this, it is possible to quickly locate any required node.

Cluster variations

As noted before in section 3.2.3, clusterings are available for the gene data. For when a user wishes to focus on a single cluster, we have provided a simple option to hide certain clusters, or select all nodes in a cluster. In this way, the user is not required to manually select all nodes in a cluster in order to work with them.

3.3.2 Multiple selection

So far, we have only described how a user can select a single node. This method alone is not practical, since it is slow to select multiple nodes. Also, in the

current state of the program it is impossible since compound selection is not available. A method will have to be available to easily select multiple nodes at once. In this section we will describe the design of a new type of multiple selection tool.

We have chosen to try a new direction with the multiple selection tool. Traditionally, in 2D a bounding box is used. A user clicks and holds a button at a point, defining one of the corners of a rectangle, then drags the input device to a different location, and at some point releases the button, defining the second point. All items within this rectangle are then selected. This principle can be easily extended to 3D, defining a box instead of a rectangle. This method is also currently implemented in SARAgene. But in virtual reality, this method poses a few usability problems. One of them is that in a VR view of a 3D network, the user may have problems noticing whether all nodes that are to be selected are actually inside the box given a limited position of the viewpoint (the user can't walk freely around the box without changing its size). In the SARAgene implementation, the problem is further enhanced by not showing any depth information about the box, its visual representation consists of only six planes that are blended with the scene.

For this project, we have implemented a bounding box on different principles that solves these problems (For full information of the evaluation results, see chapter 5). The principal change with respect to the earlier mentioned method is that the box remains in the virtual space. Instead of temporarily showing the box when the dragging starts, then hiding it again when the dragging ends, there is a menu option to show or hide the box. Once the box is visible, the user can manipulate it to encompass the nodes that should be selected, and finally a menu option is provided to actually make the selection.

The principles of the visual representation can be seen in figure 3.1. It is possible for the user to intersect one of the corner handles, and drag this around to change the size of the box. Because the box does not disappear after dragging, the user is free to change the size, walk around to see if the selection is correct, and make adjustments as necessary. Also, the edges of the box are displayed as solid lines making it easier to determine the position and size of the box in 3D space.

Chapter 4

Implementation

4.1 Introduction

In this chapter, the details for the implementation of what was described in chapter 3 will be described.

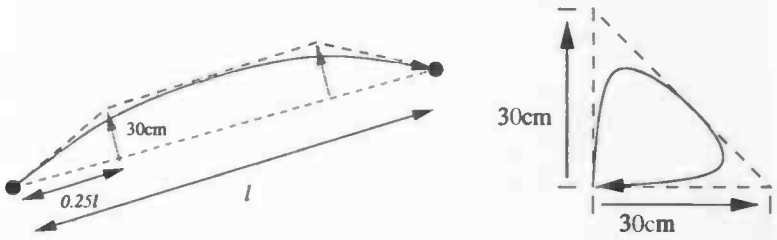
4.2 Layout

As said in the last chapter, we use the force-directed layout algorithm to compute the graph's layout. The force-directed layout uses a physical model to compute such a layout. The nodes are seen as particles repelling each other, and edges are seen as springs that try to keep the nodes they are connected to close to each other. The algorithm then tries to find a minimum-energy state for this system, and this tends to result in good layouts. Our current implementation uses a basic force-directed algorithm with the change that the spring forces are computed using a logarithmic function for the spring's length instead of the traditional use of Hooke's law.

4.3 Visual elements

We explained in section 3.2.2 that we use Bezier splines in certain situations for the lines between nodes. The control points for these splines are determined using the positions of the nodes on each end of the edge, together with a flag that controls whether the bend direction of the spline should be flipped. This is used to ensure that the two edges do not overlap. For a loop, a different method of determining the control points is used. See figure 4.1 for how the control points are determined. The values mentioned in the figure are chosen to make sure a bent edge does not deviate too far from the straight line between its two endpoints, and that a loop does not become too large.

The visual element of the node itself is very simple. The pyperbonus library provides a class named TextBoard which is a set of rectangles textured with individual letters to form any required text. The drawback of using this class is that we are limited in the colours of the text and background by the font textures available to us.



(a) Bent edge (b) Loop

Figure 4.1: Determining the control points for non-straight edges

Finally, a new class was written to represent a single node. This contains the visual element for the node itself and all its outgoing edges.

4.4 Clusters

For clustering, we need to modify the layout algorithm to put nodes in the same cluster close together. The force-directed layout algorithm makes this easy. What we do is that for each cluster, we add an *attractor node*. The positions of these nodes are initialised at the average of the positions of all genes in that cluster. We then add a spring between this attractor and all nodes in the cluster belonging to it, and add an electrical repulsion between nodes not in any cluster and the attractors. Finally, attractors repel each other, but are not influenced in any other way. This scheme causes nodes within a cluster to draw towards each other, while the clusters as a whole separate, thus making the differences between the clusters clear.

Some screen shots of the application showing the network can be seen in figures 4.2 and 4.3. Images of clustered layouts can be seen in figures 4.4 and 4.5.

4.5 Interaction

4.5.1 Highlighting

This feature was very simple to create. As noted before, all outgoing edges are stored within the node class, so all that needs to be done is iterate over all edges and change the colour and width.

4.5.2 Selection

When the network object gets an event that a node was intersected (together with the identifier of that node), we mark the node as selected. To show this selection, we initially create two text elements: one with the unselected font, and one with the selected font. This is necessary since the text class doesn't allow changing the font texture after it was created. Whenever the selection

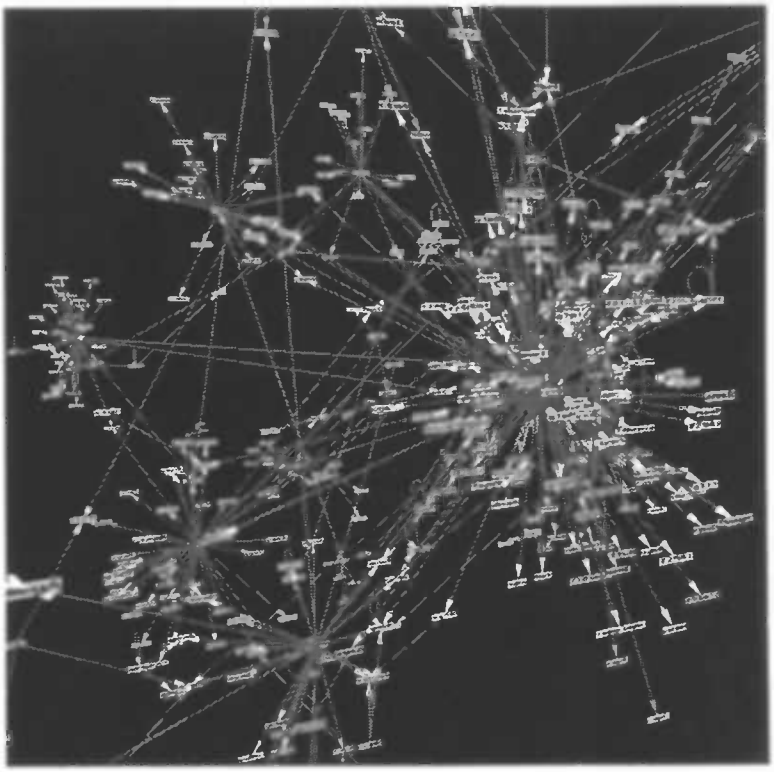


Figure 4.2: Overview of the network

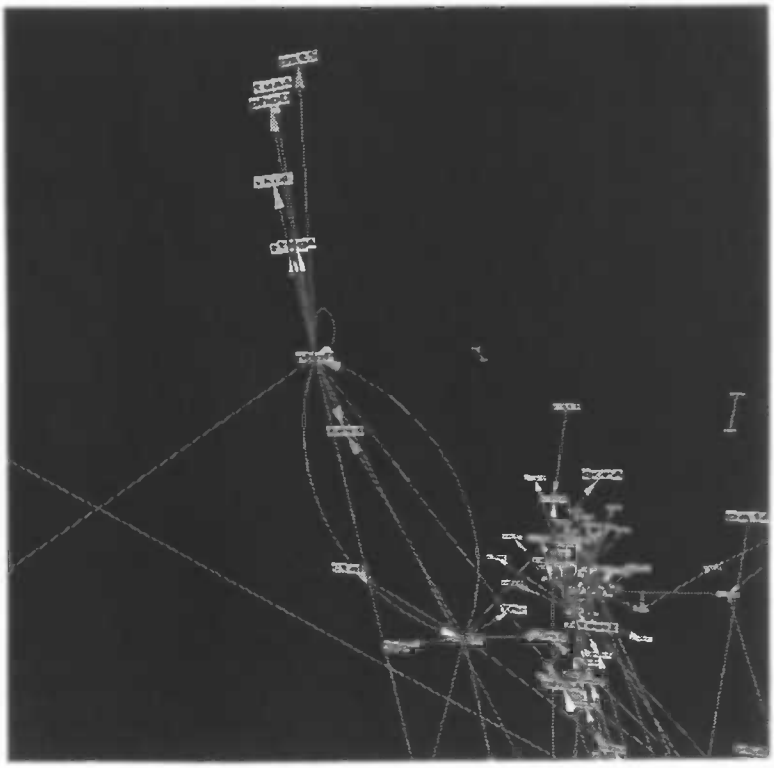


Figure 4.3: Detail showing loops and multiple edges

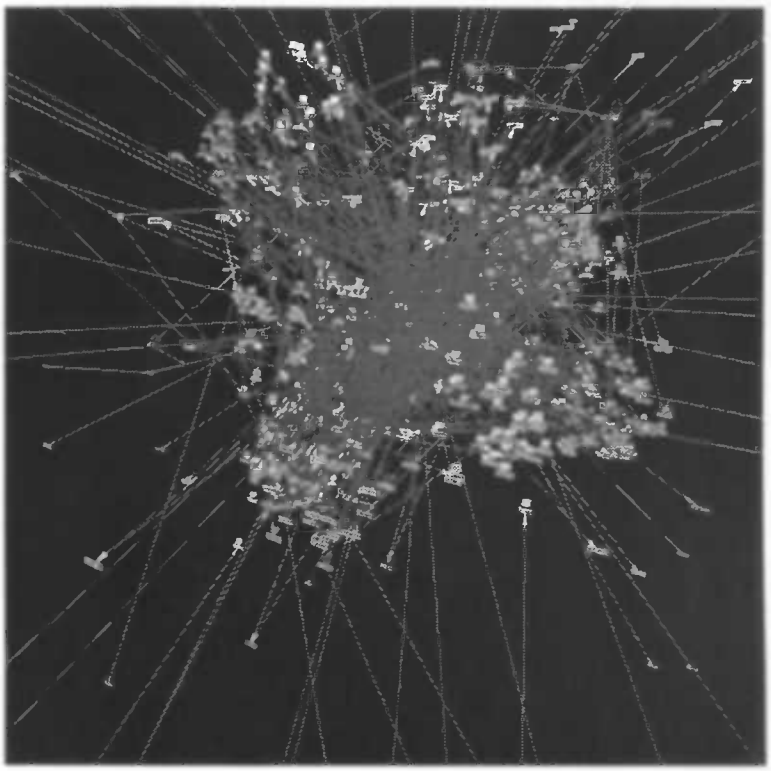


Figure 4.4: Overview using clustered data (8 clusters)

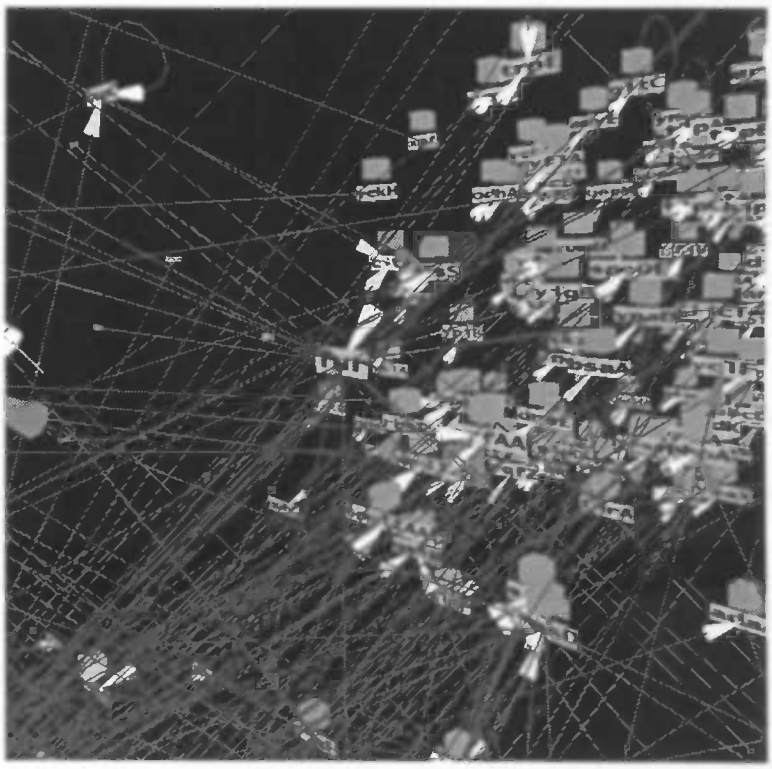


Figure 4.5: Closeup

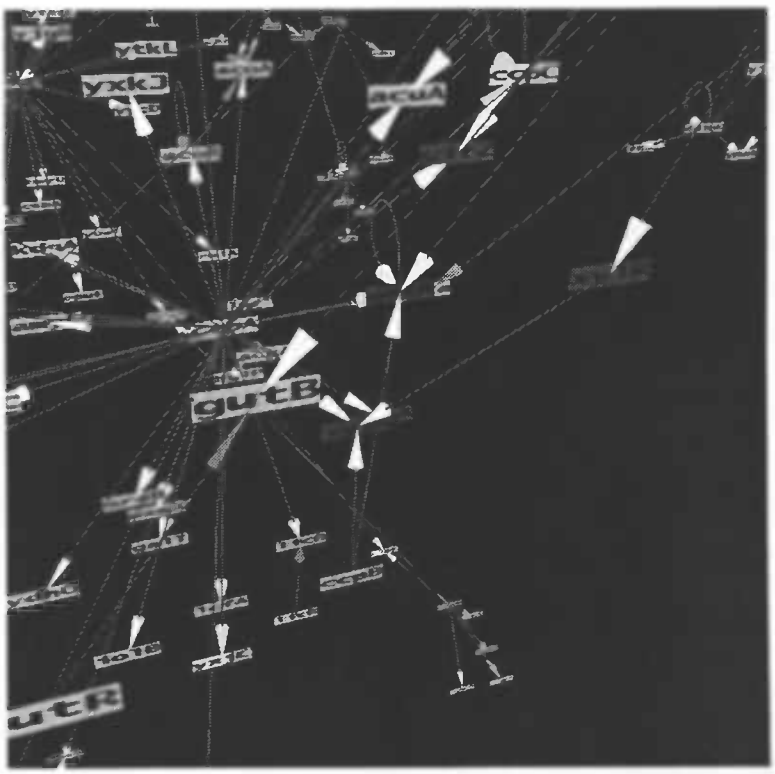


Figure 4.6: Selected nodes

changes, the relevant text element is shown and the other one is hidden. See figure 4.6 for an example where some nodes have been selected.

While selection of this type was already present in the MINT map view, due to the structure of SARAgene this had to be added manually. See section 6.1 for a more detailed review of this.

4.5.3 Filter mode

To implement this feature, we made it possible for each node to hide itself. For all filter-related functions, we can then iterate over the relevant set of nodes and set their visibility to what is required. See figure 4.7 for an example of the filter mode in the application.

4.5.4 Searching

The search box and scroll list with node names are available from SARASIM. Once a node is clicked, a modify selection procedure is called to make the selected node the only selected one. Then, the position of the node in the network is queried with another function we wrote. We can then use a linear animation class from SARASIM to move the selected node to the correct position in the CAVE.

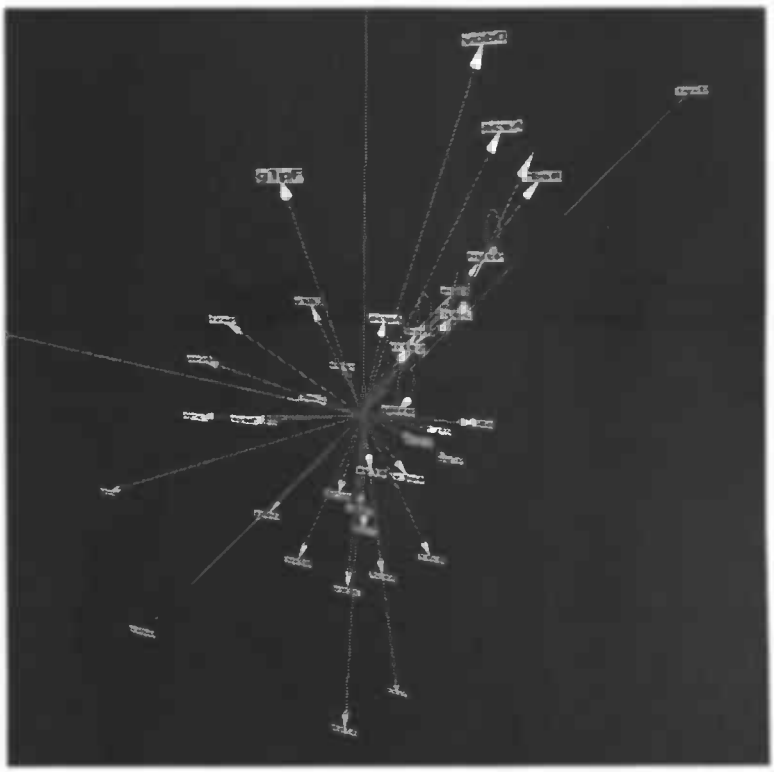


Figure 4.7: Filter mode active

4.5.5 Multiple selection

For the multiple selection box, we derived a class from the original SARAgene selection box. The reason for this was that the original class contained the procedure to determine the geometry that was inside the box, and deriving from this class allowed us to reuse this procedure. The downside was that the original class was written in C and not all (relevant) methods were public. For example, when creating the static selection box, two sets of geometry are created - one for the new box, and one for the old box. Reusing the old geometry wasn't possible since those variables weren't accessible from within Python. Also, to determine the elements within the box, we are actually enabling and then disabling the old selection box again. The rest of the box implementation is fairly straightforward.

See figure 4.8 for a screen shot of the multiple selection box active in the application.

Figure 4.8: The new **multiple** selection box

Chapter 5

Usability tests

5.1 Introduction

When creating an application, it is important to make sure that the interface will allow users to efficiently and comfortably use the application. If visualisations and interface elements have been used that are well-known, performance can be estimated from previous experience, but when creating new elements it is necessary to perform evaluations.

In this project, we have created a few new major elements: the visualisation of the DBTBS network, and the multiple selection box. Since the network visualisation, although new, is in several aspects similar to the MINT map component we evaluated in another study [14], we could use the results of that evaluation to enhance the DBTBS network visualisation. As such we have chosen not to evaluate the network, and focus on the multiple selection box instead. In this chapter we will describe the evaluation process and discuss the results.

5.2 Evaluation description

5.2.1 Method

In our evaluation, we have used the testbed evaluation method as described in [4]. Testbed evaluation is a useful method to determine the effectiveness of virtual environment interaction methods. In a testbed, interactions are broken up into sub-tasks, for which several methods may be available. This is specified in a taxonomy. A taxonomy for a testbed evaluation is a hierarchical decomposition of tasks. Major tasks are broken down into subtasks of possibly several levels, and for each lowest level task, we specify several methods that can accomplish that task. For example, the task "modify an object's colour" can be broken down into "select an object" and "select a colour". For the colour selection, we can then specify methods such as RGB sliders, or picking from a fixed palette. The taxonomy for our case can be seen in figure 5.1. In this taxonomy, only the properties which we actually tested are mentioned.

With a taxonomy created, we can pick different options and combine them

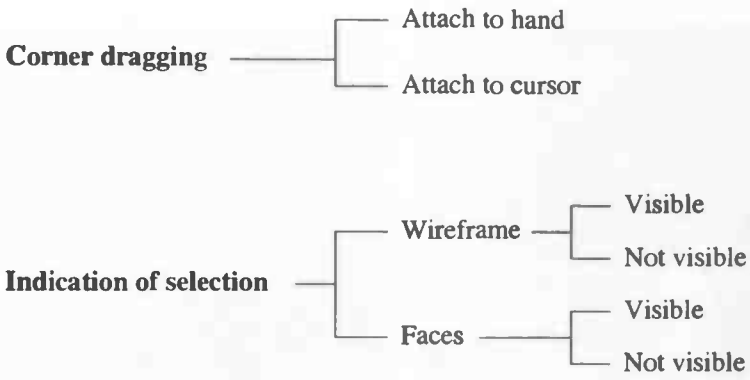


Figure 5.1: Taxonomy for our evaluation

into a setup that will be tested. By testing several combinations, we can then estimate the performance of the individual options and use this to determine the best selection of options to use.

5.2.2 Test facility

The evaluation was performed in the CAVE™ facility of the High Performance Computing and Visualisation Centre of the University of Groningen. The visualisation is done by an SGI Onyx 3400. This is a shared-memory computer with 16 CPUs and 20 GB of memory.

The input device used by participants is a wand. This is a 3D mouse that is being tracked to determine its position and orientation.

5.2.3 Evaluation procedure

We invited 15 subjects for this evaluation. Since the tool does not require any specific knowledge, we asked various people, with various amounts of experience in the CAVE. All participants were male, and university or academy students or staff. All but three participants had some prior experience with using a program in the CAVE, although in most cases this was not more than a few hours.

Since none of the participants had any previous experience with the application, they were first instructed on the basics of the tools and features necessary to successfully participate. After this they were given some time to familiarise themselves with the tools. Then, the actual tasks began.

Participant groups

Each participant was placed into one of four groups. Each of these groups had some settings changed to measure the effect on performance of selection. The settings were as follows:

- Group 1 had a selection box with visible faces and edges (as could be seen in figure 4.8). When dragging the corner handles, they would snap to the position of the wand (see figure 5.4(a) for a visual indication of this).

(a) Task 2 (b) Task 8

Figure 5.2: Examples of selection tasks

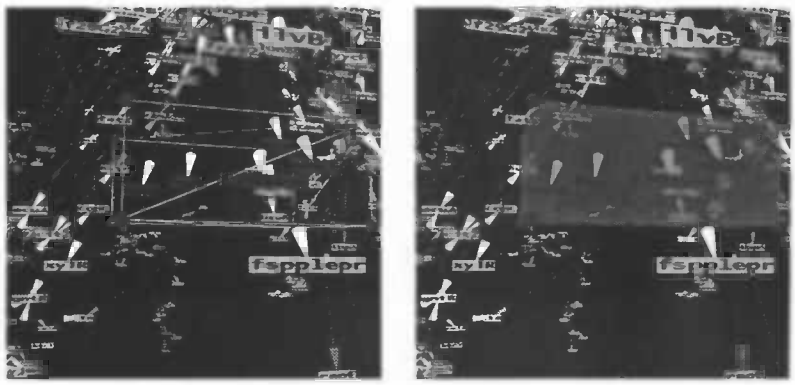
- Group 2 had the same visual settings for the box, but when dragging the corners, they would remain attached to the cursor at the point they were when the dragging started (figure 5.4(b) demonstrates this).
- Group 3 had completely transparent box faces, so only the wire-frame of the box was visible (this setup is visible in figure 5.3(a)). The corner dragging was the same as with group 2.
- Group 4 had the Faces visible, but the edges were hidden (see figure 5.3(b)). But, because of the intersection highlighting code in SARAgene the wires could become visible by intersecting the box. Corner dragging was again the same as with group 2.

Tasks

The tasks were all similar: the participant was asked to select a given set of nodes located somewhere in the network. A total of nine sets of nodes were given. These nine sets were divided into three levels of density: on the first level, no other nodes were close to the set, whereas on the highest level many nodes were close to the set requiring a more precise selection. For a list of all tasks, refer to appendix A In figure 5.2 two selection tasks are shown to give an example of a low and a high density level.

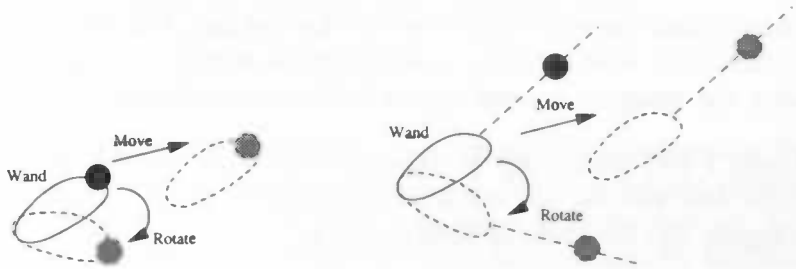
Evaluation

During the evaluation, we paid attention to two criteria: execution time and error rate. Execution time was counted from the point that the participant activated the selection box (they were asked to deactivate it between selection tasks), up to the point that they successfully selected the given set of nodes. Error rate was determined by counting the number of nodes that should have been selected, but were not, and the number of nodes that were selected, but



(a) Wire-frame-only box (b) Faces-only box

Figure 5.3: Other settings for the multiple selection box



(a) Attach to hand/wand (b) Attach to cursor

Figure 5.4: Effects of moving and rotating the wand with different dragging methods

shouldn't have been. Tasks done in-between selections (such as locating one of the nodes in the set via the search tool) were ignored.

Finally, after the evaluation the participant was given a questionnaire. This questionnaire contained two parts. The first part consisted of a number of questions answered with a 5-score *Likert scale*. This covered several aspects, not limited to the tool itself. The second part contained three essay questions in which participants could express their opinions on the tool in more detail. See appendix B for the questionnaire.

5.3 Results

5.3.1 Error results

The average error results for the four groups can be seen in figure 5.5. In table 5.1 all error statistics are shown. What is quickly noticeable is that task one was done nearly error-free, and that in task four the most errors of all tasks

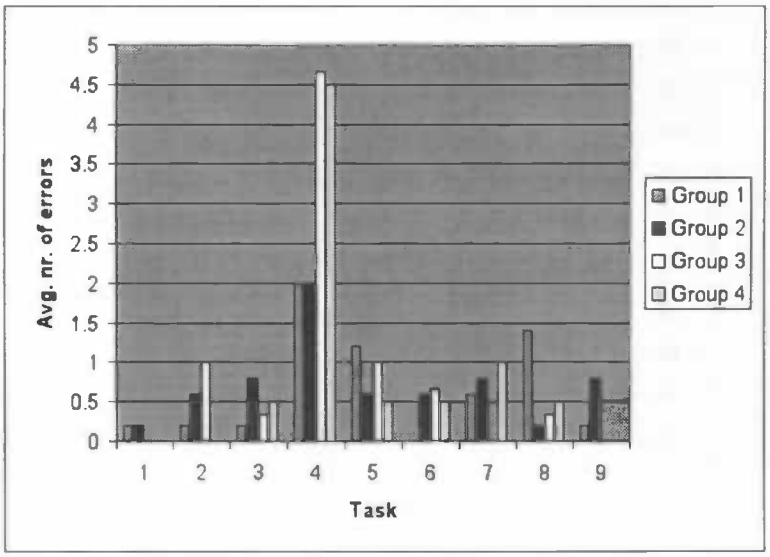


Figure 5.5: Average error count

	Group 1			Group 2			Group 3			Group 4		
Task	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
1	0	1	0.2	0	1	0.2	0	0	0.0	0	0	0.0
2	0	1	0.2	0	3	0.6	0	3	1.0	0	0	0.0
3	0	1	0.2	0	3	0.8	0	1	0.3	0	1	0.5
4	0	4	2.0	1	3	2.0	0	12	4.7	4	5	4.5
5	0	5	1.2	0	3	0.6	0	2	1.0	0	1	0.5
6	0	0	0.0	0	3	0.6	0	2	0.7	0	1	0.5
7	0	2	0.6	0	2	0.8	0	0	0.0	0	2	1.0
8	0	5	1.4	0	1	0.2	0	1	0.3	0	1	0.5
9	0	1	0.2	0	3	0.8	0	0	0.0	0	0	0.0

Table 5.1: Error results for all groups

were made. The results for task four are explained by the fact that it was the largest selection (both in the number of nodes and physical size). Since many nodes had to be selected, it was not always completely clear if all requested nodes were inside the box.

On the results of the errors, it's hard to determine which group performed the best. While group 1 performed very well in the first four tasks, the results for the last five tasks vary a lot. Also, if we compute the average number of errors per person, there is not much difference. Group one has the lowest value for this (6), and group three the highest (7.67). From this it seems that the different settings of the selection tool do not seem to influence the number of errors made. This can be ascribed to most participants checking to make sure the box encompassed all required nodes before clicking the select button.

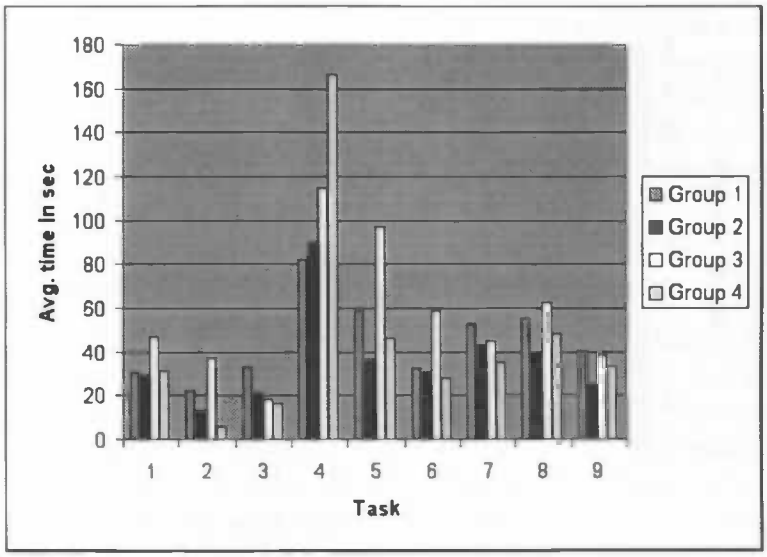


Figure 5.6: Average task completion time

5.3.2 Timing results

In figure 5.6, we can see the average timing results we obtained for the four groups. For all timing statistics, see table 5.2. We can quickly notice a few trends. The first three tasks were performed faster than the last six, and task four took by far the most time. This can easily be explained by the selections that were required in the various tasks. In the first three tasks, the nodes to be selected were pretty much free from other nodes, so the selection box could be fairly wide around the target selection. In task four (the largest selection), there was of course more time required to correct the errors. Next to that, more time was also required for a user to check whether the selection is correct. The other selections were smaller, but other nodes were close to the selection so the box had to be positioned rather precisely.

Analysing further, we can see that on average, groups two and four had low completion times (even taking into account the number of errors), while groups one and three generally took longer to make a selection. Group three especially took very long, since they did not make many more errors. This can be explained because having only a wire-frame box reduces how clear it is which nodes are in the box and which are outside. Group one, while making relatively few errors, generally required more time (only group three required more time for a few tasks). Based on the results so far we choose the settings from groups two or four as preferred settings for practical use.

5.3.3 Feedback results

As said before, feedback consisted both of a number of multiple-choice questions and a number of essay questions. The results of the multiple choice feedback for the four groups can be seen in figure 5.7. A number of non-selection related questions were also included in this set. The complete feedback questionnaire

	Group 1				Group 2			
Task	Min.	Max.	Avg.	Std. dev.	Min.	Max.	Avg.	Std. dev.
1	12	55	30.60	15.53	13	53	29.25	19.30
2	15	42	22.40	11.28	8	19	13.25	28.41
3	20	58	33.40	14.47	20	25	21.75	2.70
4	38	131	82.20	38.60	41	188	90.25	59.76
5	33	120	58.60	35.70	23	45	36.25	48.53
6	29	37	32.40	3.78	19	56	30.50	15.02
7	38	83	52.60	17.98	30	63	43.00	12.51
8	29	127	55.40	40.73	17	66	39.00	20.02
9	27	70	40.20	17.51	16	36	25.50	17.85
	Group 3				**Group 4**			
1	34	57	47.00	11.79	21	41	31.00	14.14
2	10	75	37.30	33.71	5	7	6.00	1.41
3	9	29	18.00	10.15	11	22	16.50	7.78
4	58	215	114.70	87.13	78	254	166.00	124.45
5	55	139	97.00	59.40	38	55	46.50	12.02
6	20	100	59.00	40.04	27	29	28.00	1.41
7	35	55	45.00	14.14	33	38	35.50	3.54
8	27	128	62.30	56.92	19	77	48.00	41.01
9	31	53	38.70	12.42	24	42	33.00	12.73

Table 5.2: Timing results for all groups

can be seen in appendix B.

The multiple-choice questions addressed various areas. This includes both the tool specifically, the network itself and immersion.

Looking at the results for selection, it becomes clear that groups two and four rated their settings for selection the best, and group three rated it the worst. Thus, the groups that had settings that made selecting harder (judging from the errors and timing results), also considered the selection tool to be less useful or comfortable. Overall satisfaction is fairly similar through all groups, with the groups with higher ratings for selection also rating the overall satisfaction higher. None of the groups seemed to have problems with disorientation or immersion sickness.

The essay questions allowed the participants to write down what they liked about the tool, what they disliked, and what they would like to see changed. This helps identify the strong and weak points of the tool, and helped us determine the optimum settings of the tool as well as some future improvements.

In the free feedback, some things were noticeable. The most common suggestion was that there should be some kind of immediate indication that shows which nodes were in the box and which ones weren't. Also, group 1 suggested that the way the box corners were dragged was changed to the method used by the other three groups. Also, some users had issues with the box position being fixed to the CAVE rather than the network. This was an issue when for example the user accidentally (or on purpose, to move the network to a better

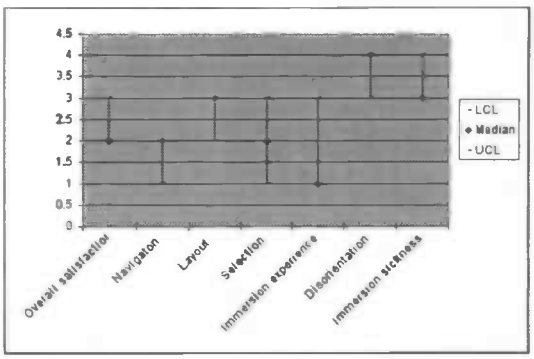
position) hit the joystick on the wand and moved the network as a result - extra effort was then needed to move the box to the relatively same position again. Alternatively, an option to move the box as a whole instead of just one of the nodes is an option. Finally, a zoom option was suggested. This tool would grow or shrink the network in place, making it easier to either select large areas, or small selections close to other nodes.

5.3.4 Analysis

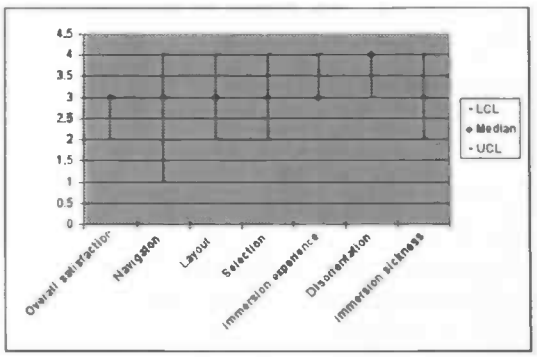
From the results shown earlier a few conclusions can be drawn. First of all, as we expected, having completely transparent faces (group 3) drastically decreases performance. Second, the method of dragging corners used by group 1 is generally considered unpleasant and also is not the most efficient method. Overall, we conclude that using the settings from group 2 is the best option: a partially transparent box with visible lines, and dragging the corners by linking them to the cursor. Although performance and user feedback from group two and group four is almost equal, for group 4 the visualisation changes often due to the highlighting. We think that having a visualisation that does not change is preferable.

5.3.5 On searching

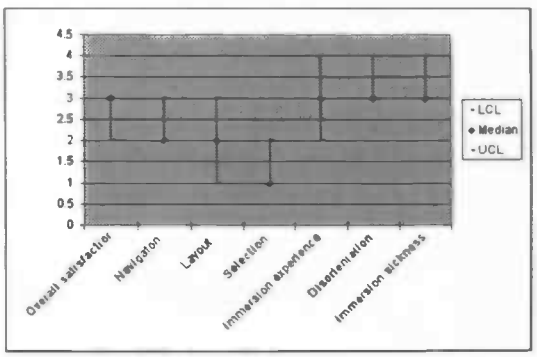
Although the search feature (see section 3.3.1) wasn't officially tested, one point should be made. While evaluating the MINT application in SARAgene, it became very obvious that the search feature as it was implemented there was not very useful: some participants in that evaluation simply failed to locate the node they searched for. We identified a few reasons for this. First, the view was not scrolled so that the selected node was in the centre of the CAVE. Instead, the view placed the selected node in the centre of the floor. Secondly, the node was not highlighted in any way, requiring reading of the labels to identify the node. These issues were taken into account and the search feature for the DBTBS application acted differently from the MINT version as a result. While performing the selection evaluation, it became clear that the modifications greatly helped: every participant was able to locate the requested node within seconds of the view coming to a stop, even a node that was partially hidden behind another node.



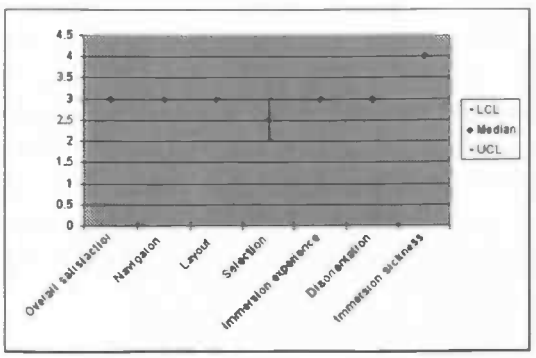
(a) Group 1



(b) Group 2



(c) Group 3



(d) Group 4

Figure 5.7: Feedback results for the four groups

Chapter 6

Conclusion

In this thesis we have described a visualisation method for gene regulatory networks, and the details on the implementation of this visualisation in the SARAgene application. Also, we can use this visualisation in a slightly modified form to show the network when taking into account clustering information for the nodes of the network.

The normal network visualisation gives a fairly clear overview. Although there are some areas with a high density of edges, these are not very common, and even there the edges do not significantly obscure any node labels. In the case that a part of the network is distracting, it is also possible to hide uninteresting nodes. A small issue in the current layout algorithm has to do with separate subgraphs. Since nothing special is done for such subgraphs they will be pushed away from the largest subgraph in a random direction, which can make it hard to locate them visually.

When using clustering information to create a layout, the network as a whole becomes fairly cluttered. Filtering is still possible and helps comprehension though.

Also, we have described a new type of selection tool for use in virtual environments. We evaluated the effectiveness of this tool and determined under what settings users were able to use it in the most efficient way.

6.1 SARAgene as framework

Before we can determine how well SARAgene works as a framework for genomics applications in VR, we must first determine what a framework should consist of. In [23] it is said that a framework is a collection of concrete and abstract classes, and that it should promote reuse. Based on this, we have to conclude that SARAgene makes a poor framework. To illustrate this, we'll use experience from this project.

For the addition of the network view, two new components were needed: one data encapsulation object, and one graphical view object. For the dataset, there is no base class. Of course, with the many different types of data sources available, this is understandable. For the graphical view, it's different though. What SARAgene provides is a class called `Visual`, and this is the class from

which all visualisations should be derived. It provides some basic functionality for changing the placing of the geometry, and registers itself with a *Connection object*, which facilitates exchanging the locations of the same item (gene, molecule, etc.) in different visualisations so that the connections between these items can be shown. Beyond that, there is no added functionality on top of SARASIM. This is one of the reasons that in our implementation of the regulatory network visualisation, even basic interaction such as selection had to be added manually. Although source files for other views provided a good basis, there was still the need to customise the selection code for our view.

In short, while SARAgene is presented as a framework, it is just an application. If this is to be turned into a framework, significant features will have to be added to ease the creation of new data sources and data views.

6.2 Future work

There is still a lot of room for improvement in the program. We will discuss some of these improvements in this section.

6.2.1 Cleaner graph view

Although we already offer features to reduce the complexity of the displayed graph, these features consist of hiding a part of the graph. This makes it difficult to examine interactions over longer distances. One method that the application may greatly benefit from is described in [21]. In this method, the user has a local region of interest (ROI) that can be moved around. Within this ROI the full detail of the graph is displayed. Outside the ROI, dense clusters of nodes are combined into single, larger nodes. This greatly reduces the complexity of the graph while maintaining a global overview, and the user can simply move the ROI around the graph to view the details in a different section.

Also, the earlier mentioned issue of separate graphs needs to be addressed. This will involve identifying separate graphs, performing the layout algorithm separately on each of these subgraphs and then placing them closely to each other in a regular fashion.

6.2.2 Speedups

The application in its current form suffers from low execution speed at a number of elements. One of these is when a different cluster method is selected. Since the layout changes can be quite large, the user may become disoriented if the change is immediate, so we added a smooth transition. Unfortunately, doing a single frame update of this transition takes more time than we have determined the transition should last, nullifying the effect. Another issue is the layout algorithm. Originally this was written in Python along with the rest of the application, but after some tests we determined that computing the layout of a network on the Onyx would take roughly 45 minutes.

These problems are due to several factors. First, Python is an interpreted language, and these kind of languages execute slower than natively compiled

executables. Second, a single processor of the Onyx at our facility is not particularly fast, adding to the problem of low execution speed. Finally, the Python interpreter is not thread-safe, which makes it impossible to divide the computational load over the different processors available. Some kind of solution needs to be found to alleviate these problems and help make some features of the application practical for interactive visualisation.

6.2.3 Selection enhancements

While the user has the ability to select both single nodes, and multiple nodes contained within a box, the selection features end there: it is not possible to add to or subtract from the current selection. Mainly this is due to the lack of buttons. On normal PC systems, modifier keys on the keyboard are employed for additional selection modes, but in our CAVE setup, we are limited to the wand, of which all buttons are used by SARAgene.

Also, as was determined in the evaluation of the selection box, a clear indication of which nodes are inside the box should be added.

6.2.4 Billboards

Currently, the node names are displayed on text boards. These boards are static relatively to world space, which means that when a user looks at it from the side, the board will show up as a line. Bill-boarding is a method in which the boards are rotated to face the viewer, solving this problem. While bill-boarding is available in SARASIM, this has the unwanted side-effect of text boards not responding to selection events anymore. This will need to be improved.

Bibliography

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular biology of the cell.* Garland Publishing, Inc., New York, NY, USA, second edition, 1994.

[2] Giuseppe Di Battista, Ioannis G. Tollis, Peter Eades, and Roberto Tamassia. *Graph Drawing, Algorithms For The Visualization of Graphs.* Prentice Hall, 1999.

[3] John Bohannon. The human genome in 3d, at your fingertips. *Science*, 298(5594):737, 25 October 2002.

[4] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. Testbed evaluation of virtual environment interaction techniques. In *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 26–33. ACM Press, 1999.

[5] T. A. Brown. *Genetics: a molecular approach.* Chapman & Hall, London, 1999.

[6] CAVELib: `http://www.vrco.com/CAVELib/OverviewCAVELib.html`.

[7] J. A. Dickerson, Y. Yang, K. Blom, A. Reinot, J. Lie, C. Cruz-Neira, and E. S. Wurtele. Using virtual reality to understand complex metabolic networks. *Atlantic Symposium on Computational Biology and Genomic Information Systems and Technology*, pages 950–953, September 2003.

[8] C. Healey, K. Booth, and J. Enns. Harnessing preattentive processes for multivariate data visualization. In *Proceedings Graphics Interface '93 (Toronto, Canada, 1993)*, pages 107–117, Toronto, Canada, 1993.

[9] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, /2000.

[10] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[11] B. Lewin. *Genes VII.* Oxford University Press, New York, 2000.

[12] Yuko Makita, Mitsuteru Nakao, Naotake Ogasawara, and Kenta Nakai. DBTBS: database of transcriptional regulation in bacillus subtilis and

its contribution to comparative genomics. *Nucleic Acids Research*, 32(Database-Issue):75–77, 2004.

[13] Henrik Rojas Nagel, Erik Granum, and Peter Musaeus. Methods for visual mining of data in virtual reality. In *Proceedings of the International Workshop on Visual Data Mining, in conjunction with ECML/PKDD2001*, Freiburg, Germany, September 2001. 2nd European Conference on Machine Learning and 5th European Conference on Principles and Practice of Knowledge Discovery in Databases.

[14] P.J. Ogao, M.P. Visser, O.P. Kuipers, B. Stolk, P. Wielinga, and J.B.T.M. Roerdink. Visualization of genomics information: A usability case study of SARAgene virtual reality application. 2005.

[15] OpenGL|Performer: http://www.sgi.com/products/software/performer/.

[16] Python programming language: `http://www.python.org`.

[17] Igor Rojdestvenski. Metabolic pathways in three dimensions. *Bioinformatics*, 19(18):2436–2441, 2003.

[18] Bram Stolk. Collaboration on the SARAsim platform, a guide for users and developers. 2003.

[19] Bram Stolk. SARASIM - a modular approach to virtual reality. 2003.

[20] Bram Stolk, Faizal Abdoelrahman, Anton Koning, Paul Wielinga, Jean-Marc Neefs, Andrew Stubbs, An de Bondt, Peter Leemands, and Peter van de Spek. Mining the human genome using virtual reality. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 17–21. Eurographics Association, 2002.

[21] Frank van Ham and Jarke J. van Wijk. Interactive visualization of small world graphs. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, pages 199–206. IEEE Computer Society, 2004.

[22] P. C. Winter, I. Hickey, and H. L. Fletcher. *Instant Notes In Genetics*. Oxford, 1998.

[23] Rebecca J. Wirfs-Brock and Ralph E. Johnson. Surveying current research in object-oriented design. *Commun. ACM*, 33(9):104–124, 1990.

Appendix A

Task description

Task No.	Benchmark Task	Time taken to complete task	No. of errors made on task
	Spend some time to familiarise with the multiple selection tool. You will be asked to select nodes at three varying levels of proximity to each other.		
	LEVEL 1 - Low density selection		
1	Select nodes: fNR, narK, narG, ywiD		
2	Select nodes: clpP, clpE, cplX, lonA, ctsR		
3	Select nodes: pabA, traP, trpE, yhaG, ycbK		
	LEVEL 2 - Medium density selection		
4	Select all nodes influenced by node 'fur'		
5	Select nodes sigW, abh and ydbS		
6	Select nodes ccpA, xylA and yjmA		
	LEVEL 3 - High density selection		
7	Select nodes: roK, comGA, comC and comEA		
8	Select nodes: spoOA, sdh and sinR		
9	Select nodes: spoOH, dppa and f82-156		

Appendix B

Feedback questionnaire

Please check the appropriate circle that most correctly reflects your impression concerning SARAgene-DBTBS.

1	Rate your overall satisfaction with the multiple selection tool:				
	O Very dissatisfied	O Dissatisfied	O Neither satisfied nor dissatisfied	O Satisfied	O Very satisfied
2	Rate the effectiveness of the wand controls in navigating the graph:				
	O Very ineffective	O Ineffective	O Neither effective nor ineffective	O Effective	O Very effective
3	How useful was the layout for the network:				
	O Not very useful	O Not useful	O Neither useful nor not useful	O Useful	O Very useful
4	Rate the use of the wand for selection:				
	O Very hard	O Hard	O Neither hard nor easy	O Easy	O Very easy
5	Did you feel as if you were a part of the environment (as if you were a physical entity in the application):				
	O Felt like an observer	O Not really	O Not so sure	O Somewhat	O Very much a part
6	Did you get disoriented in the network:				
	O Always	O Most of the time	O Not so sure	O Some of the time	O Never
7	Did you at any time during the evaluation experience some dizziness, sickness, or became uncomfortable:				
	O Very uncomfortable	O Uncomfortable	O Can't say	O Comfortable	O Very comfortable

What do you like most about the multiple selection tool?

What do you like least about the multiple selection tool?

What would you like to see implemented/refined in the tool?