User Interaction Discovery in Virtual Environments

Student Name: L. A. Sutton

Supervisor Name: W. Li

Submitted as part of the degree of BSc. Natural Sciences to the

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

April 28, 2015

***Abstract* —**

Context/Background In the past 20 years there has been an rapid increase in computer mediated interactions between people. There has been much previous work examining visualising networks of people in these virtual environments such as social networks or within office communication systems, however this has almost exclusively focused on static systems of relationships rather than a representation of the dynamic interactions themselves.

Aims What has been learned from the visualisation of static networks of relationships is to be applied to the representation of dynamic systems with a focus on the interactions between the users rather than simply the static relationships. This will be used to produce a new visualisation system that will allow previously obscured features of the data set to become apparent.

Method A system has been created in Python using the Pygame library to represent various different modes of interactions between users in a virtual environment in different ways along with the state of the users. This system has then been applied to various models of real-world scenarios in order to demonstrate its utility.

Results The system is able to effectively represent several different real-world scenarios across a wide variety of situations for example the spread of viral advertising, the propagation of adoption of behaviours or a trust-based recommendation system. it can then be seen that this system of looking at interactions is more effective than currently available approaches in several areas. The solution then effectively represents these situations in ways clearer than would otherwise be possible.

Conclusions In conclusion, the results of this solution meet its aims. The application to various models show how it effective to represent systems in terms of the interactions between users and how these interactions evolve over time. It can then be seen how this is a more useful way of looking at these systems that simply a static graph of relationships.

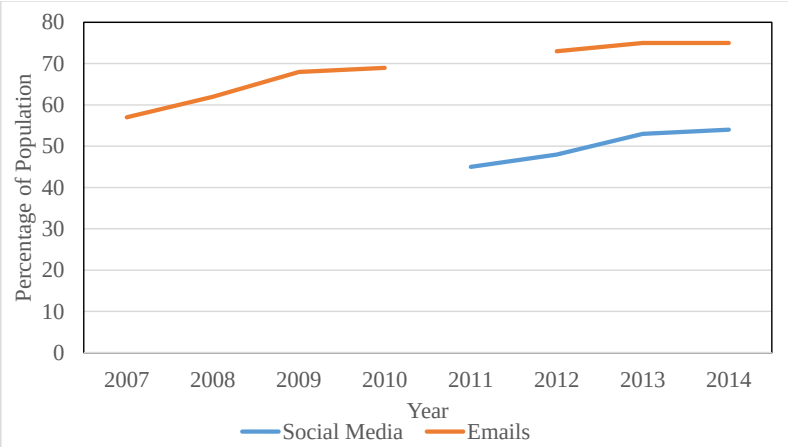
***Keywords* —** user interaction; virtual environments; visualisation; clustering

I INTRODUCTION

In the 21st century, people spend more time than ever interacting in virtual environments. Figure 1 shows how social media and email in particular are not only very commonly used in the UK, but ever growing in popularity. Indeed they are both used by the majority of the population. With this growth it is clear that it is becoming ever more important that these systems of relationships and interactions can be effectively understood not only though statistical analysis, but through effective visualisation.

There is a long history of visualising the structures that form within these environments as graphs (Freeman 2000). This has evolved significantly over time from early graphs of relationships drawn by hand to the present day where layout algorithms running on computers make it possible to visualise ever more complex graphs with greater numbers of nodes effectively.

Figure 1: Graph of growth on online interactions in the UK (Office for National Statistics 2014)



Finally, as an extension of these graphs, interactivity has become possible such that they are able to be manipulated in real time. This allows increasingly novel ways of visualising data and opens up the possibility that these graphs can be used to show more than simple, static relationships but rather become effective tools for visualising a dynamic system of interactions that evolves over time.

In this project I have taken 'User Interaction' to mean any way in which users consciously, intentionally affect others. This could take place over a period of time of be instantaneous; it could be between two users or many; it could be a single event or it could be ongoing. All of these different ways of interacting are called 'modes'. For example I will consider interactions such as users sending emails between one another but not interactions such as a user's advertising preference changing what another user sees.

Virtual environments in this project will mean any environment in which users are able to interact in the ways previously mentioned, mediated by computers. This will most likely mean over the internet, but it could also be over smaller network such as within an office or university internal network. It may also be taken to mean a 3 dimensional virtual environment in which users can interact as avatars. This could be for example a video game, a social networking website or a messaging system such as email.

A Project Motivation

Previous attempts to visualise the structure of virtual environments through relationships between their uses have suffered two main drawbacks: Firstly they have only been interested in static 'snapshots' of networks that either represent a state at a particular time or use a static graph to represent a period of evolution. This is a result of the ability to produce dynamic representations being relatively new in an already well established field of research. Secondly they have focused only on the relationships between people within these networks rather than exploring the interactions themselves that make up these relationships. In some cases the relationships that are shown are inferred from interactions and in others the interactions are ignored completely and the relationship data gathered in a different way.

With increasing ease of interacting in virtual environments and the explosion of modes over which these interactions are possible it is increasingly important that new ways are developed to

visualise this data. It was therefore decided that current systems could be extended in two ways: Firstly to take what was learned from static representations of relationships and apply this to dynamic systems of interactions that were able to evolve over time. Secondly to move the focus of the systems away from the current paradigm of representing only the relationships between users that are inferred from the interactions and instead represent the interactions themselves.

B Project Aims

In order to guide the project, a set of deliverables were set out that the project aimed to meet. These evolved over time as the project was in progress and were broadly split into basic, intermediate and advanced objectives.

The basic objective was to create a system for modelling user interaction with a visual output. This model would be based on an existing system and the visual output should be able to handle more than one mode of interaction

This was then to be extend through an intermediate objective. First the basic system should be evolved so that it was able to include changes of state of the users within the model affecting the way in which they interact. The system should then also include a way of visualising clustering of users according to their reactions and finally the system should allow real-world data to be used as an input for the models.

Finally the system was to be extended through advanced objectives. In order to meet these the system should be adapted so that changes in the way user interactions cluster should be visualised and this should be able to be examined quantitatively. Finally this system should be shown to be applicable to real-world problems.

II RELATED WORK

A Social Network Visualisation

As has already mentioned, network visualisation is a science with a long history. Here, however the focus will be on the period after computers begun being used to assist in the drawing of social network graphs. There have been created low-level tools that exist with the intention that they have the necessary flexibility to accommodate a wide variety of visualisation styles and techniques (Heer et al. 2005). As well as this, there are tools that exist to provide numerical analysis (Borgatti et al. 2002). However, these tools have focused on the analysis and visualisation of snapshots of data remaining static, rather than data sets that evolve over time. They therefore demonstrate the problems that I have talked about above.

These have been used to build ways of visualising social network data such as that from Friendster (Heer & Boyd 2005) in order to facilitate easier information discovery than was previously possible, showing that there is merit in using a graphical approach, and indicating that another approach may be useful. Systems intended for use by members of the public also exist (Wolfram Research, Inc. 2015) which again reinforce this and show that this is something that need not be difficult to use.

B Information Presentation

Much previous research has been carried out into the presentation of data. The most popular model can be summed up as 'Overview, Zoom, Filter' (Shneiderman 1996). In this model it is

suggested that data should be presented as a movable field of view with emphasis on allowing the user to gain an 'overview' of relevant data and identify areas which will be of interest. Specific areas of interest can then be brought into focus preserving the context of the overall picture before extra information is viewed. Also shown is the importance of smooth display updates and responsiveness to user input, something considered in this project. Building on this is the concept of distorting the 'presentation space' (Carpendale & Montagnese 2001). This is the method used in Vizster and is common usage in many other data visualisation applications. It imagines that the virtual space in which the data is presented is a real material that can be stretched and viewed through a movable lens as necessary to make the relevant areas of the visualisation more clear. These ideas can expanded on by comparison of lenses and mathematical frameworks for implementation (Leung & Apperley 1994). Together these papers can provide a unified way of presenting graphs in a dynamic way and this is the approach that will therefore be taken in this project.

Work has also been completed on evaluation of data presentation. Most notably it is possible to find sets of criteria for effective data presentation (Tufte & Graves-Morris 1983). These can be then used in the evaluation of this specific implementation.

C Graph Drawing

There is also previous literature on the drawing of graphs in aesthetically pleasing ways and so as to maximise their ability to convey information. The most effective ways for drawing graphs such as this is my making use of a force-directed algorithm (Himsolt 1995) because of the effectiveness in displaying the structure of the graph clearly despite is drawbacks in terms of speed. In this algorithm, each node is modelled as repelling each other node and the edges between them are modelled as springs (Fruchterman & Reingold 1991). However, this algorithm doesn't scale well with rapidly increasing numbers of vertices. It has been pointed out that with a large number of nodes, calculating the layout in this way is very expensive in terms of computing power. With the correct optimisations it is possible to reduce the complexity to $o(nlog(n))$ (Barnes & Hut 1986). Unfortunately, this significantly increases the complexity of the implementation. A compromise is therefore taken in this solution whereby there is a limit on the number of nodes that can be laid out for a given application.

D Modelling Network Interactions

The ability to produce networks of relations and interactions from many different data sources is also explored in a variety of different papers. For example, networks of social interaction have been produced from a history of email correspondence within an organisation (Fisher & Dourish 2004) or the transcript of an internet relay chat (Mutton 2004). These explore problems associated with inferring complete graphs from incomplete data. Alternatively, it is possible to use data gathered from a users's social network using tools made for the purpose of downloading such data in a usable form. For example the Netvizz Facebook app (https://apps.facebook.com/netvizz/), which allows the download of a text file containign a list of a users's 'friends' along with various attributes about them and a list of connections between them.

There are also many models that have been proposed for interactions in other ways. These include models for the spread of influence to adopt behaviours (Kempe et al. 2003), the spread of viral advertising (Van der Lans et al. 2010) and a proposal for a social recommendation system

(Walter et al. 2008). The combination of all of these gives many possible systems which can be used to demonstrate the effectiveness of the system.

E Categorisation of Interactions

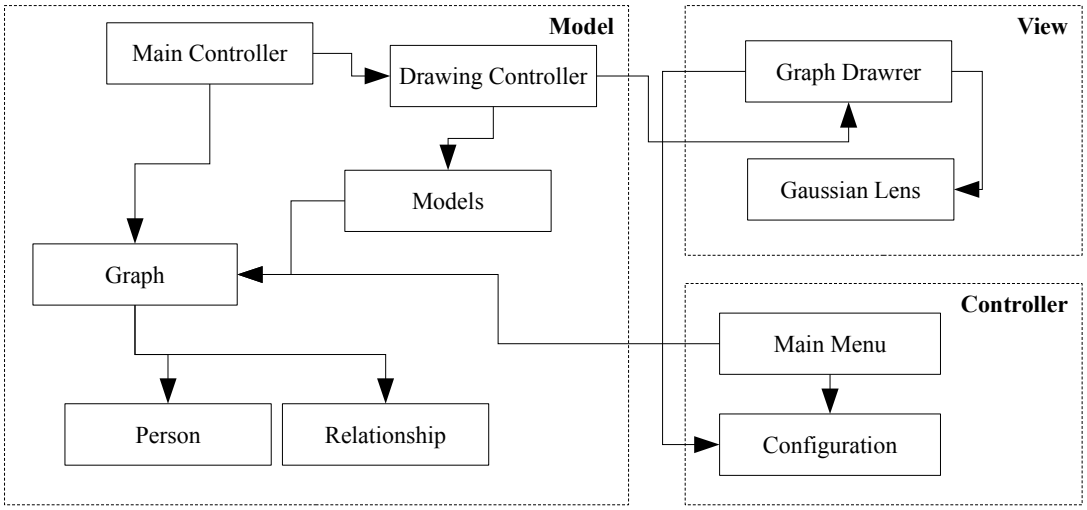
Previous research has also explored categorisation of interactions by their characteristics. This has mainly in the past been applied to social iterations within 3D virtual environments in which people interact as avatars, referred to as Networked-Virtual Environments. One particular application of this is games (Manninen 2000), where we can see that there are multiple ways of categorising interactions, but most usefully using communicative action theory to categorise interactions by their purpose. This has the advantage that it can categorise interactions across modes but the drawback that it can be difficult find interactions of the same purpose across different environments. This is complimented in by comparing interactions in a selection of game environments (Becker & Mark 2002). Extending this to other environments such as the social network, it can be shown how much of the interaction that goes on within a virtual environment is hidden from the user. For example Facebook collecting information about us including our interactions which changes our experience in ways in which might not initially consider meaningful (Schneier 2010), it is therefore important that we consciously exclude or include those in our visualisations.

III SOLUTION

A Architecture

A 'Model, View, Controller' software pattern has been used to guide the architecture of the solution. This has allowed the components to remain as separate as possible which was important so that the visualisation could remain the same between different data sets being visualised and different configurations being used.

Figure 2: Architectural diagram of solution



It can be seen in figure 2 how the the Model contains all necessary logic in the system along its state: The graph contains the state of the users and the relationships between while the drawing controller provides an interface to the View and the Models handles the logic of evolution of the data. This is kept separate from the Controller which handles the Menu generation and storage of its results as Configuration objects. The View is then again separated and interacts with the Model and the Controller to draw the graph and distort it with the Gaussian Lens.

B Menu System

The system is primarily controlled though a system of menus which are presented to the user before the solution begins running.

Figure 3: Example menu system of the solution

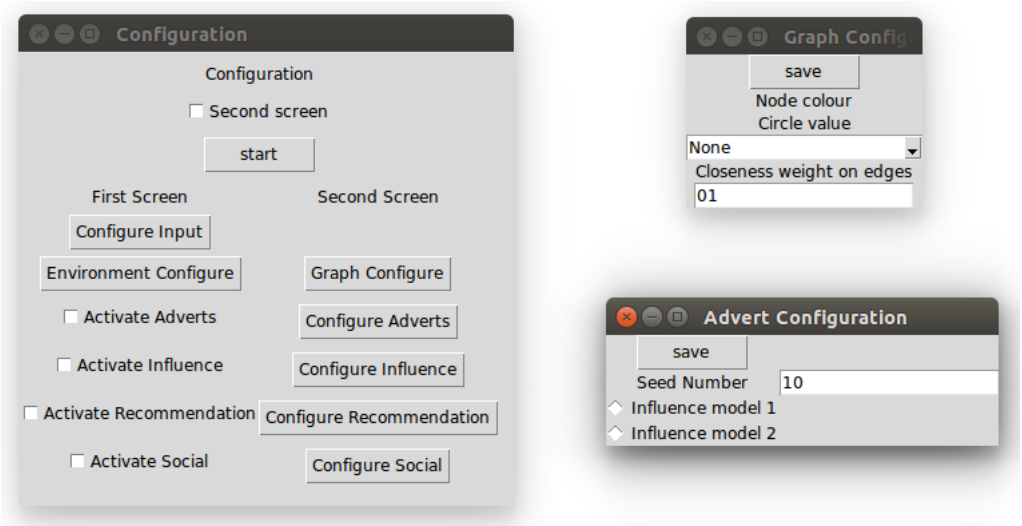


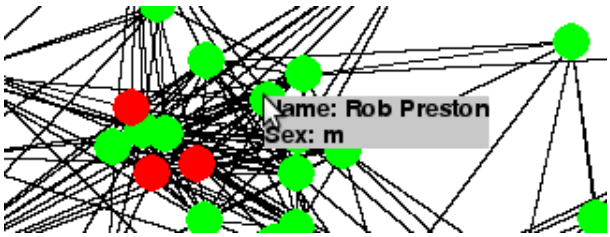
Figure 3 shows an example of these menus. The main window on the left represents the main control of the application. In this window it is possible to enable and disable various models that can be used to provide data for the visualisation system. The buttons then allow various aspects of each model to be configured. The aspects that can be configured are detailed in the descriptions of each system below, we see examples of this in the menus on the right.

This menu also allows the user to select what initial input data will be used. They can either generate random data with limited realism or import their own. Users own data will come from the Netvizz Facebook app (https://apps.facebook.com/netvizz). This allows users to download a '.gml' file containing a list of all friends of a user along with various of their attributes and a list of all friendships between the listed users in order to build up a network.

C Interactions with Vertices

The user is able to interact with vertices representing people in the graph in two different ways as shown in figure 4. The first way is that they have the ability to move their mouse over a node in order to see more information about it. This can be seen as the box with a grey background in the figure. It can be configured depending on what information is desired.

Figure 4: Diagram showing labels and selctions



The second way is by clicking on them. This can be used when it is necessary to choose vertices, for example it could be used to select the points being used to seed the advert or influence models laid out below, the colour change is then used to indicate which vertices are selected.

D Graph Layout

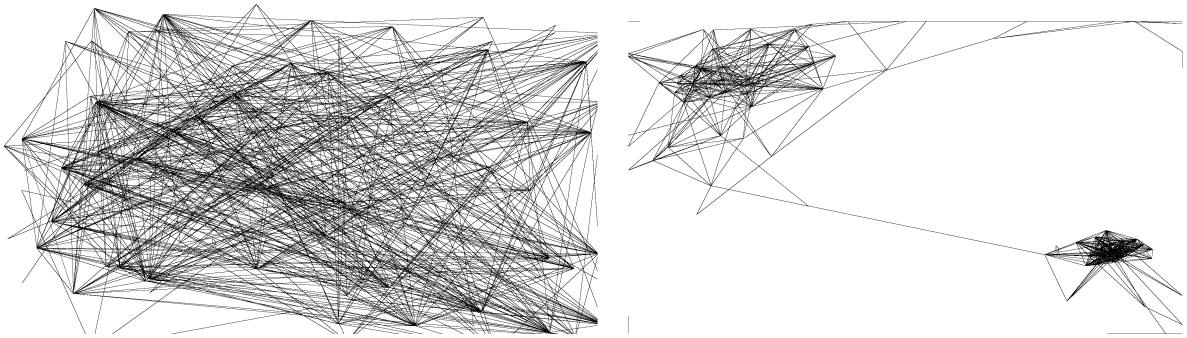
This element allows the system to lay out the graph, forming the basis for its representation of interactions. Current data visualisations make almost exclusive use of a force-directed spring layout for graph drawing as described above. These springs, as in my case, need not respond to force in the same way as a real, spring but can instead have whatever response gives the best layout for the graph.

Algorithm 1 Force-directed spring layout algorithm

$graph \leftarrow (V, E)$ ▷ Graph represented as a collection of vertices and edges
$k \leftarrow \sqrt{1/|V|}$
$t \leftarrow 0.05$
for all Vertex in V **do**
  $Vertex.displacement \leftarrow (0, 0)$
  for all Other in {V\Vertex} **do**
    $dist \leftarrow distance(Vertex, Other)$ ▷ $distance$ gets Euclidean distance
    $diff \leftarrow (Vertex.x - Other.x, Vertex.y - Other.y)$
    $Vertex.displacement \leftarrow Vertex.displacement + diff \times (k^2/dist)$
for all Edge in E **do** ▷ Edge between two vertices, V_1 and V_2
  $dist \leftarrow distance(V_1, V_2)$
  $diff \leftarrow (V_1.x - V_2.x, V_1.y - V_2.y)$
  $V_0.displacement \leftarrow V_0.displacement - diff \times (dist^2/2k)$
  $V_1.displacement \leftarrow V_1.displacement + diff \times (dist^2/2k)$
for all Vertex in V **do**
  $(Vertex.x, Vertex.y) \leftarrow (Vertex.x, Vertex.y) +$
    $(Vertex.displacement/distance(Vertex.displacement)) \times$
    $(min(distance(Vertex.displacement), t))$ ▷ min gives minimum of two elements
  $(Vertex.x, Vertex.y) \leftarrow min(0.95, max((Vertex.x, Vertex.y), 0.05))$ ▷ max gives maximum of two elements

The algorithm uses an iterative approach in order to find a local point of least tension on the

Figure 5: Example of application of graph layout algorithm



springs and is based on (Fruchterman & Reingold 1991). I have detailed my implementation as Algorithm 1.

This is split into three main parts after the set up. where the value of k is set to the value suggested in the above paper and t is set to a value which was determined through experimental testing. Every vertex starts with 0 displacement. The algorithm then calculates the repulsion between every pair of vertices and updates its displacement proportional to the inverse of the distance. The algorithm then calculates the attraction along every edge and updates the displacement of each vertex in this edge proportional to the square of the distance. Finally, each vertex is moved either its displacement or t, a pre-determined distance, whichever is smaller, then the algorithm checks that none of the points have been moved outside the boundary of the screen.

In my implementation, the input graph is initially laid out totally at random. We can then see an example of an application of this algorithm over a number of iterations in figure 5. It can be seen on the left how the points are initially given a random layout with only the edges between them drawn and it is impossible to see any structure in the graph. After approximately 10 iterations of the algorithm the image on the right is obtained where elements of the graph become clear showing two main clusters, one much tighter than the other.

E Lens

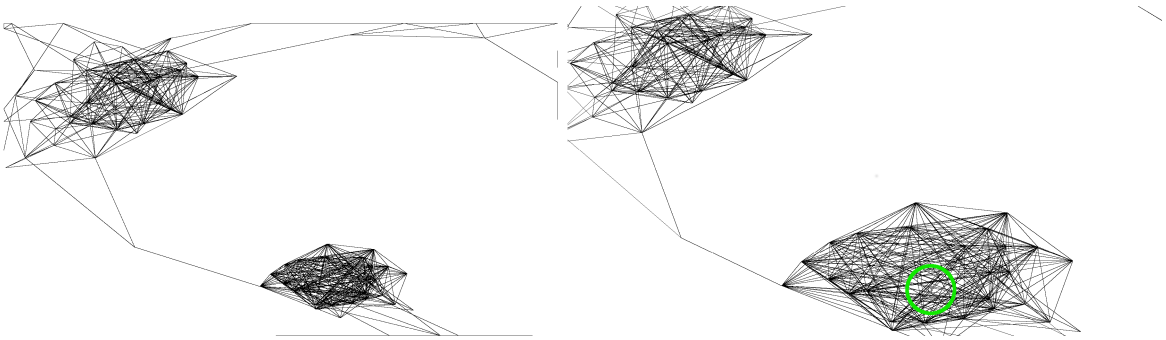
Details of graphs, especially when they include a large number of nodes, can become hidden. After experimentation it was determined that a Gaussian lens would be best suited to solve this problem because of its gentle falloff and smooth transition through the point of maximum magnification as the user moves the 'lens' around the visualisation. Gauss's equation is given by

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}. \tag{1}$$

In my solution $f(x, \mu, \sigma)$ represents the distance that the point will be displaced away form the cursor. $x - \mu$ is the initial distance between the vertex and the cursor and σ is a value that was determined by experiment in order to give the best result. 0.1 was chosen, meaning that vertices over 20% of the width or height of the screen away would no longer be significantly affected.

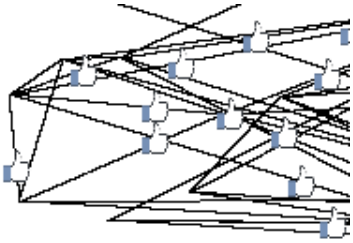
An example of the lens in use is shown in figure 6. Here on the left we see an image with no lens effects, on the right we see the image affected by the lens centred on the cursor (highlighted by a green circle). It can be seen how it is much easier to see the structure of the tightly clustered group of points in the bottom right through the use of this lens.

Figure 6: Example of use of Gaussian lens



F Transitions

Figure 7: Examples of transitions



One way of showing interactions between users in real time is by showing movement in the graph. An example of this can be seen in figure 7. This example might be used in a social model to show 'likes'; As a user likes another's post on a social network, a 'thumb' image is generated at the person who likes and travels over the course of about a second to the person who's post is being liked. In other models the 'thumb' can be replaced by another image or by a simple disc or other relevant symbol.

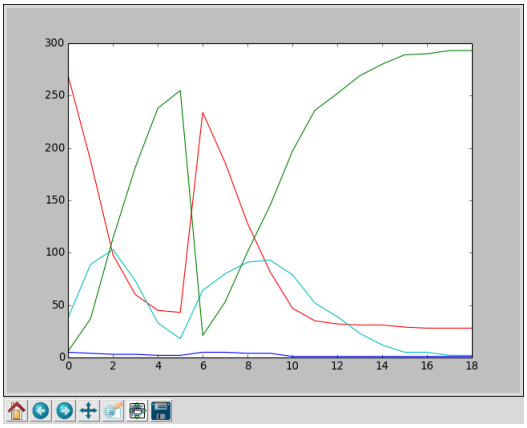
G Quantitative Output

It is possible for the system to record the state of each user at given times and then to display this data once the model has run. This both makes it easier for a user to make sense of a rapidly evolving model and allows the possibility of a quantitative output for later analysis.

An example of this can be seen in figure 8. The horizontal axis represents time while the vertical axis represents the number of users. In this case the colours in the graph match the colours that are used to represent the vertices in the network.

This visualisation makes it clear how the distribution of users changed over time. In this view the progress of an advertising campaign can clearly be seen by the jump when this campaign is ended and a second introduced. This compliments watching the evolution in real time allowing different features to be seen.

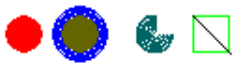
Figure 8: Example if a display of the history of states



H Vertices

The system allows many different ways of representing people as vertices depending on their attributes in order to make it as clear as possible. Figure 9 shows examples of different ways that attributes of the vertex can be displayed in order to show different information.

Figure 9: Examples of possible representations of vertices

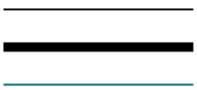


The vertex on the left represents the most basic with one solid colour, this could change between preset colours to represent changes between a finite number of states or could have its brightness or hue changed to represent a continuous change in a property. The second vertex from the left shows that this can be build up in layers with an edge or another band of colour. Each layer can then be used to represent different properties of the same vertex. The second vertex from the right shows the possibility of representing the properties by the length of an arc. This could be used to represent the state of a property that it is known will fall between two bounds for example age. The final example on the far fight shows how it is possible to represent vertices as arbitrary shapes constructed as necessary in order to be fully customisable for whatever is being represented at the time.

I Edges

As with vertices, there are multiple ways of using edges of the graphs to represent properties of the graph. A selection of these can be seen in figure 10.

Figure 10: Examples of possible representations of edges



The top line shows how edges are generally represented as a single pixel line joining the two users. The middle line then shows that the thickness can be changed to represent properties of the relationship such as strength or interaction frequency. On the bottom I have given an example of colour change for the line. This could be represent some interaction between he users and can be combined with either of the two previous methods. Finally it is also possible to add significance to the length of the lines by adjusting the graph layout algorithm. This could then be used to affect the shape of the graph in order to show more complex interactions between properties.

J Clustering

My model is able to calculate the global clustering coefficient C, defined as

$$C = \frac{number\ of\ closed\ triplets}{number\ of\ triplets\ of\ vertices}. \tag{2}$$

This provides a measure of the degree to which nodes in the graph are interconnected with one another and so its effect on models such as influence spread can clearly be seen.

K Demonstration Models

In the solution a set of demonstrations have been developed in order to show off the usefulness and effectiveness of the implementation as applied to real-world scenarios.

K.1 Influence

Two established models of influence spread through a social network (Kempe et al. 2003) were implemented. This could be used to model behaviour adoption across a social network, for example if a new social game were to be released, this could simulate how users encouraged their friends to also play the game.

In the first model is based on a 'Linear Threshold'. Initially each vertex is given an adoption value, representing how much influence must be exerted over them in order for them to adopt the behaviour. A set of starting nodes are then chosen to seed the behaviour and the model then proceeds in discrete steps. At each step, if the relationship strengths of vertices that has adopted divided by the relationship strengths of all neighbours is greater than the vertices adoption value, that vertex then adopts the behaviour.

The second model of influence is based on an 'Independent Cascade'. A set of seed nodes are chosen to adopt the behaviour, then in steps each vertex that has adopted the behaviour will have one chance to influence its neighbours to adopt the behaviour. Whether or not they are successful depends partly on the strength of the relationship and is partly random.

In this model it is possible to configure many different aspects such as the number of starting nodes and the relative chances of the behaviour being adopted. It is also possible to use both models simultaneously in order to compare their results.

K.2 Advertising

A simulation of the spread of viral advertising was implemented based on a current model (Van der Lans et al. 2010) which relies on a transition of each person between a number of

states. These states depend on whether they have or havent participated, or in what way they have been asked to participate.

In the initial set-up, a random number of 'seed' emails are sent to people, then a number of 'seed' adverts are shown, both with a chance to trigger participation in the campaign. As before this proceeds in steps. At each step, a certain number of vertices will check their email. If they have received an email related to the campaign, either from another user or a 'seed' email, this will give them a chance to participate. If they choose to participate, they will then generate a normally distributed number of emails to their friends about the campaign and the cycle continues. The system as implemented here also allows new campaigns to be started while current ones are in progress and they can then run in parallel, independently.

In this case it is possible to change all of the relevant variables such as the number of seed emails sent and the number of users initially seeing the seed advert. It is also possible to determine how likely a user is to respond to either an advert or an email not necessarily at the same rate.

K.3 Recommendation

Finally, I implemented a model of social recommendation system (Walter et al. 2008). This model imagines a system in which users are recommended something, say, films according to what their friends report they enjoyed.

This system begins by seeding a number of ratings for a number of different items. Recommendations are then propagated through the graph. This happens once, each vertex looks through its neighbours to see if it can find someone who has a direct experience of the film, if it fails to find anyone it then looks through its neighbours' neighbours and so on until it either finds someone or reaches a pre-determined depth. If it does find one or more people it then takes this recommendation with a degradation depending on the degrees of separation.

It is possible in this model to change several variables such as the number of users that are initially seeded with experiences of what is being recommended, what experience these users have and the depth to which a user will search for a recommendation. This system also accommodates there being multiple items which can each have their own recommendations propagated independently.

L Tools Used

The solution is implemented using a combination of Python 3.4 and various libraries. A 64 bit binary of version 3.4.2 of CPython was used as the interpreter (www.python.org). This was chosen as it was the latest version of the most popular Python interpreter. I used a 64 bit edition so that if it became necessary I would be able to make use of all available memory and python 3 was chosen rather than python 2 so that any recent performance optimisations could be utilised.

Visualisations were produced using the PyGame library. This was obtained this by building the source (https://bitbucket.org/pygame/pygame) with CPython from above. Pygame was chosen because of its ease of use over OpenGL assisting with rapid prototyping. Additionally its use of optimised C code would ensure that the visualisations would not interfere with time required for other computations.

Menus were implemented using the Python package Tkinter. This was obtained from my system's package repository (http://archive.ubuntu.com/ubuntu/dists/utopic/). I chose to use this

as it would provide easy implementation of menus in my solution while not distracting from the models being used.

Graph drawing was done using the pyplot library from Matplotlib. This was as with Tkinter obtained from my system's package repository. I chose this as a method of graph drawing as it provided quick drawing of simple graphs and would easily allow me to change the style in order to experiment with different ways of displaying the data.

M Verification and Validation

Software verification was undertaken at all stages of the implementation. This was primarily achieved with reference to my Design Report, in which I had given thought to the design and architecture necessary to achieve the objectives set out at the beginning of the project.

Software validation was done in reference to the objectives and functional requirements set out in my Design Report which were designed to allow the objectives of the project to be met in several layered steps. This was also assisted by the project supervisor who advised on direction at all stages and ensured that focus was maintained on the areas in which it was most needed.

N Testing

As I prototyped my implementation, testing was mainly undertaken through a dynamic approach, matching the output given by the code to an expected output based on the software design. This was carried out at all stages of implementation in order to check progress against aims and ensure that subsequent work would be able to function as expected. In the case of algorithm checking however static testing was undertaken in order to ensure that the results that they gave were as expected.

The dynamic testing used a grey-box approach with a focus on the visual output. While the system is highly modular, the modules necessarily have a high degree of interdependence in order to provide a useful output. Given that no module was particularly complex, testing was focused at an integration and system level enabling more time to be given to the output visible to the user as was the focus of the project.

O Implementation Difficulties

The main difficulty that I faced during implementation was the speed at which the Python would run using the interpreter on my computer. It was necessary at all times to make a special effort to ensure that algorithms remained as fast as possible so that the user interface would not spend long periods unresponsive as this would degraded the user experience.

There were attempts made to overcome this by using the alternative Python interpreter PyPy. However, this did not support the Pygame binaries I was using at the time and I was unable to use it to build suitable binaries. While there is an alternative to Pygame compatible with PyPy, Pygame_cffi (https://github.com/CTPUG/pygame_cffi), at the time of writing this did not appear sufficiently complete or stable for use with the project. I also tried to replace some of my loops with NumPy arrays (Van Der Walt et al. 2011). However it became apparent that the cost of initialising many of these small arrays was greater than the time saving for elimination of loops. The same difficulty was encountered while spreading the computation through multiple threads due to the relatively high cost of creating threads.

Additionally, part way though the project, Facebook changed its privacy policy, restricting the amount of data that was accessible through its API. This means that at the time of writing NetVizz is no longer able to download new data. It would be necessary therefore that an alternative solution was found in order to continue using a users's own social network data.

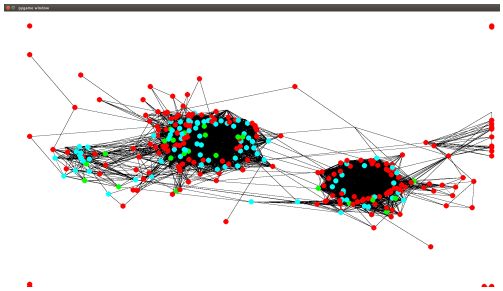
IV RESULTS

In order to demonstrate my system several mock ups were made to show that it could be effective in making use of combinations of the models and techniques I have detailed above.

A Viral System

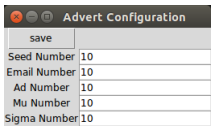
Here the viral advertising model specified above was implemented. This was then then used to demonstrate various aspects of the representation possible in my system.

(a) Viral graph view, showing the states of the people



(c) Graph showing evolution of states of people over time

(b) Window showing configuration of new viral campaign



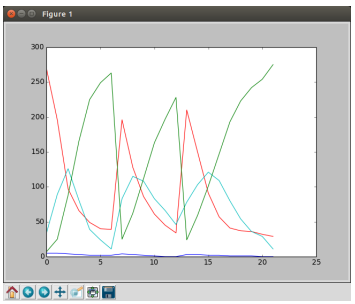


Figure 11: Examples of output produced from Viral model

In figure 11a we can see a representation of the state of all the actors in the model. In this case the colour represents the current state of the person in relation to the current campaign. It can immediately be seen that people at the edge of the network haven't yet heard about the campaign at all and are in red, meanwhile the people who have already been sent emails (in turquoise), can be seen to be clustered around the people who have already taken part in the campaign (in green).

Figure 11b shows the kind of configuration that is possible when adding a new campaign during the running of the model along with preset suggested parameters that are used by default.

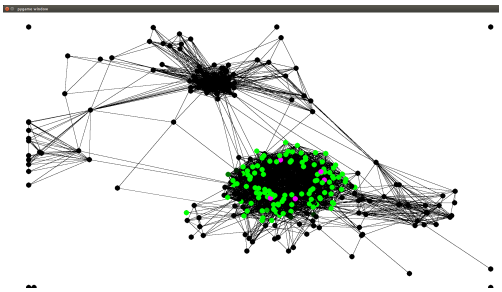
Figure 11c demonstrates how the time evolution of the model can then be shown in a quantitative way after it has run. On the vertical axis we have the number of people in each state at a given time and on the horizontal axis we have the number of iterations of the simulation. The

colours match those that were used to represent the states on the graph making interpretation intuitive. This allows important features to be picked out such as the beginning of new campaigns as well as more subtle features such as the time it takes for new participants to slow.

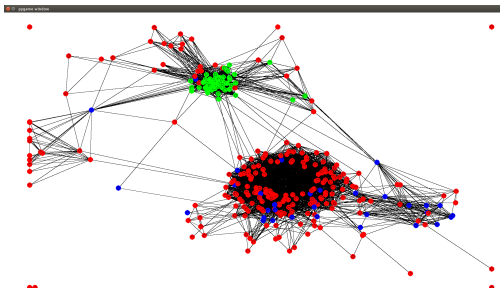
B Recommendation System

The trust-based recommendation system was then implemented in order to demonstrate a different set of features. The system was run for three products simultaneously using different sets of starting nodes for each product.

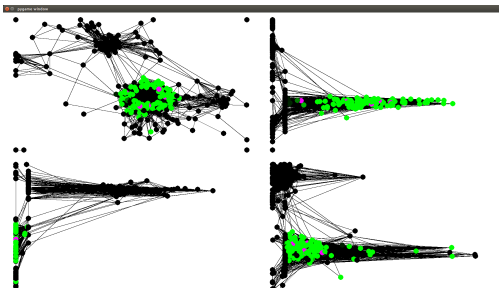
(a) Initial graph view showing who has experienced and who has been recommended a product



(b) View demonstrating everyone's favourite product



(c) View comparing what products people like



(d) View showing different distribution of preerence for another product

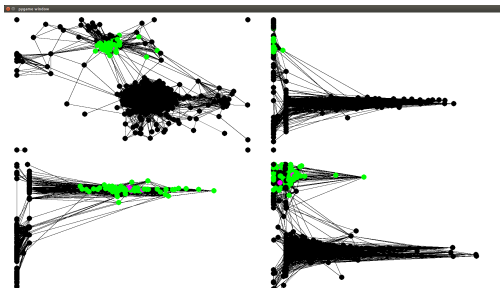


Figure 12: Examples of output produced from the recommendation model

Here we can see in figure 12a that the user is able to see the effect of those who have experience of a product on who will be recommended it. In this case those people who appear purple have a direct experience meanwhile those who appear in green are those who will be recommended the product. The brightness of the green depends on the confidence of the recommendation. It can therefore be seen that as only people in one cluster have any experience of this product, only that cluster will be recommended it.

Figure 12b then demonstrates a different way of looking at this data. The colours represent the item which each person is most likely to enjoy by making use of the three simultaneous recommendation systems. Red for one film, green for another and blue for a third.

Then in figure 12c we can see a different layout of the data. Here each of the streched horizontal polts represents one of the three simultaneous recommendation systems. The horizontal distance from the left represents how much they are likely to enjoy the product based on the system while the layout of these plots aims to keep the clusters in relationships distinguished. The

highlights show the people who have experience of and have been recommended the products as before. Figure 12d then shows different highlights, based on the second product. This easily allows anyone to see the relationships between preferences between the product and how they cluster.

C Influence

Finally the same was done making use of both models of influence spread detailed above.

(a) Showing selection of vertices to be influenced

(b) Showing comparison of two models of influence spread

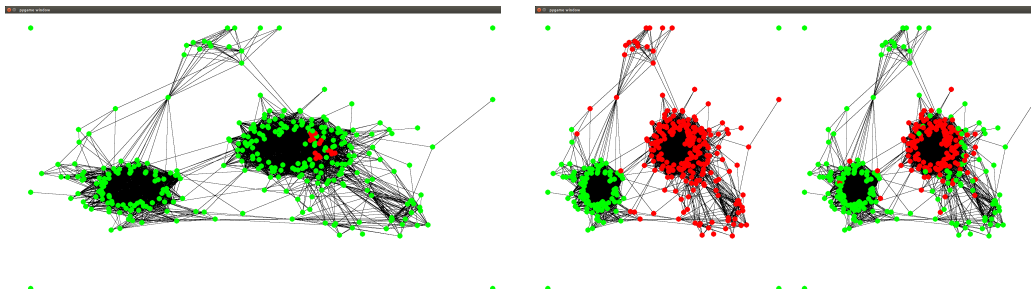


Figure 13: Example outputs form the influence package

In figure 13a we can see the operation of selecting which will be the people who are initially responsible for influencing the rest of the population. This selection occurs by clicking on the green circles which then turn red to indicate they have been selected.

We can then see in figure 13b how it is possible to compare the two different models of the spread of influence. Once a person has become influenced they turn from green to red allowing the spread from the initial points can be seen. The side-by-side comparison then means that differences between the two models can be easily identified.

V EVALUATION

The project will be evaluated according to Edward Tufte's principles of information visualisation (Tufte & Graves-Morris 1983). These represent a check-list for effective data visualisation that, although it is many years old is still often used in connection with modern graphic design.

Therefore according to Tufte, graphical representations should:

1. **Show the data**

   It is clear that my system does indeed show the data. The calculations that I perform are in order to provide the data from models, I do not process the data from these models itself and so the solution clearly fulfils this criterion.

2. **Induce the viewer to think about the substance rather than about methodology, graphic design, the technology of graphic production or something else**

   This solution is able to allow a user to focus on the substance because of its simplicity. The limited interactivity allows a user to focus on areas that they need to without becoming distracted. On top of this the graphics are simple and so do not detract from the substance.

3. **Avoid distorting what the data has to say**

   Some aspects of the system ensure that the data is not distorted, for example properties represented by a linear relationship to to a feature of the graph such as colour. This makes it difficult for the system to distort data. On the other hand in elements such the layout are unpredictable. This gives the possibility of distortion if the layout algorithm were to find an poor local minimum or if it were to be run for an insufficient number of iterations.

4. **Present many numbers in a small space**

   My solution can take up any pre-defined area, by default it produces an output the size of the whole screen. The solution also tries to ensure that in order to show the maximum amount of data it can it does this in an efficient way. However, it is optimised for certain scales rather than being able to accommodate a wide range of display sizes.

5. **Make large data sets coherent**

   This isn't something the system is able to do effectively. Due to the nature of the layout algorithm the system becomes unusably slow beyond a few hundred data points. Additionally, beyond this number the view will become too cluttered and make the visualisation unusable.

6. **Encourage the eye to compare different pieces of data**

   The system ensures that it is possible for different pieces of data to be compared. It achieves this through a side-by-side comparison as can be seen in the figures 13b and 12c. Figure 12c also shows an alternative way of comparison whereby different colours can be applied to the same layout of points in order to show differences.

7. **Reveal the data at several levels of detail, from a broad overview to the fine structure**

   The solution does not perform particularly well at this aspect as it follows a broadly flat visualisation. However, it attempts to address some aspects of this in its ability to view extra data through hovering over data points or using magnification of areas of the graph which are very tightly packed.

8. **Serve a reasonably clear purpose: description, exploration, tabulation and decoration**

   The extent to which my implementation fulfils a clear purpose depends on what it is applied to as it is designed with the intention that it could be applied to many different data sets. It can be seen in the results above though that it is possible to tailor this to specific applications, enabling, disabling and customising features as necessary.

9. **Be closely integrated with the statistical and verbal descriptions of the data set**

   This is an element that the solution does not address particularly effectively. First of all little thought was given to providing a verbal explanation of the results and in many cases the visualisations themselves have little meaning without such explanation. However, as I have explained above, generally visualisations are closely related to statistical representations of the data set.

A Project Organisation

Overall the organisation of the project was effective making use of a combination of external and self-imposed deadlines in order to ensure that work progressed week by week throughout the time given. Git was made use of to provide a version control system so that work could take place over multiple machines and to back up work to a remote server to ensure it could not all be lost.

The design of the project changed many times during the implementation and probably would have benefited from being better defined at an earlier stage however this flexibility gave the opportunity to experiment with many different techniques before settling on a final implementation.

B Layout Algorithm

One way of evaluating a graph layout algorithm quantitatively is by looking at its reduction in the number of lines that cross each other (Himsolt 1995).

Over 5 samples my layout algorithm was able to reduce crossings to a mean of 44.5% of their initial value with a standard deviation of 2.0%. This shows its effectiveness making the layout clear allowing features such as the connection of clusters to be seen that wouldn't otherwise be apparent.

VI CONCLUSIONS

Overall I believe I have met the aims of my project: I have effectively managed to produce a solution that is able to combine previous research in data presentation in social networks and apply this to a dynamic system of interactions. It can be seen that my solution is able to not only display relationships between people interacting in an environment but can as well represent the more fundamental interactions themselves. It is also able to represent elements of time evolution of those interactions rather than only a snapshot. Finally it is able to output quantitative information information that can be used along with the visualisation in order to gain additional understanding.

On top of this the models that I have implemented show that this system is a useful one that can be applied to a diverse range of systems. The three that I have demonstrated here are all very different in nature but my system is able to offer some unique way of viewing the data in each. In each case my visualisation systems offer something different to a simple static overview of the relationships and in most cases I believe makes elements of the data set clear that wouldn't otherwise be apparent particularly with regards to clustering as a result of the interactions which become obvious with certain visualisations.

Through my evaluation we can see that the solution is reasonably effective in achieving its aims to be an effective graphical representation, mostly in the areas of ensuring that data isn't distorted and its ability to ensure that implementation doesn't get in the way of substance. However it could still benefit from additional work in some areas, particularly the problems of its ability to view the data at different levels of detail and its ability had handle large data sets.

As an extension I believe that my work would benefit from re-writing in a faster language than Python which would provide two benefits. First it would allow more computation of the data to take place, particularly where data was changing in real time according to multiple models of

evolution at once. Secondly it would allow a better user experience with smoother interactions as the program was no longer waiting between drawing each frame and updating the view.

Another potential extension my work would be to provide a more cohesive interface that allows the user to interact with models in more depth as they are being run. For example the user may wish to change the way in which something is represented and didn't want to start the system again with new settings. However it must be ensured that this didn't distract with the substance of the visual output.

References

Barnes, J. & Hut, P. (1986), 'A hierarchical o (n log n) force-calculation algorithm'.

Becker, B. & Mark, G. (2002), Social conventions in computermediated communication: A comparison of three online shared virtual environments, *in* 'The social life of avatars', Springer, pp. 19–39.

Borgatti, S. P., Everett, M. G. & Freeman, L. C. (2002), 'Ucinet 6 for windows', *Harvard: Analytic Technologies* **185**.

Carpendale, M. S. T. & Montagnese, C. (2001), A framework for unifying presentation space, *in* 'Proceedings of the 14th annual ACM symposium on User interface software and technology', ACM, pp. 61–70.

Fisher, D. & Dourish, P. (2004), Social and temporal structures in everyday collaboration, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 551–558.

Freeman, L. C. (2000), 'Visualizing social networks', *Journal of social structure* **1**(1), 4.

Fruchterman, T. M. & Reingold, E. M. (1991), 'Graph drawing by force-directed placement', *Software: Practice and experience* **21**(11), 1129–1164.

Heer, J. & Boyd, D. (2005), Vizster: Visualizing online social networks, *in* 'Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on', IEEE, pp. 32–39.

Heer, J., Card, S. K. & Landay, J. A. (2005), Prefuse: a toolkit for interactive information visualization, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 421–430.

Himsolt, M. (1995), 'Comparing and evaluating layout algorithms within graphed', *journal of Visual Languages and Computing* **6**(3), 255–273.

Kempe, D., Kleinberg, J. & Tardos, É. (2003), Maximizing the spread of influence through a social network, *in* 'Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 137–146.

Leung, Y. K. & Apperley, M. D. (1994), 'A review and taxonomy of distortion-oriented presentation techniques', *ACM Transactions on Computer-Human Interaction (TOCHI)* **1**(2), 126–160.

Manninen, T. (2000), Interaction in networked virtual environments as communicative action: Social theory and multi-player games, *in* 'Groupware, 2000. CRIWG 2000. Proceedings. Sixth International Workshop on', IEEE, pp. 154–157.

Mutton, P. (2004), Inferring and visualizing social networks on internet relay chat, *in* 'Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on', IEEE, pp. 35–43.

Office for National Statistics (2014), 'Internet access - households and individuals 2014'.
URL: *http://www.ons.gov.uk/ons/dcp171778_373584.pdf*

Schneier, B. (2010), 'A taxonomy of social networking data', *Security & Privacy, IEEE* **8**(4), 88–88.

Shneiderman, B. (1996), The eyes have it: A task by data type taxonomy for information visualizations, *in* 'Visual Languages, 1996. Proceedings., IEEE Symposium on', IEEE, pp. 336–343.

Tufte, E. R. & Graves-Morris, P. (1983), *The visual display of quantitative information*, Vol. 2, Graphics press Cheshire, CT.

Van der Lans, R., Van Bruggen, G., Eliashberg, J. & Wierenga, B. (2010), 'A viral branching model for predicting the spread of electronic word of mouth', *Marketing Science* **29**(2), 348–365.

Van Der Walt, S., Colbert, S. C. & Varoquaux, G. (2011), 'The numpy array: a structure for efficient numerical computation', *Computing in Science & Engineering* **13**(2), 22–30.

Walter, F. E., Battiston, S. & Schweitzer, F. (2008), 'A model of a trust-based recommendation system on a social network', *Autonomous Agents and Multi-Agent Systems* **16**(1), 57–74.

Wolfram Research, Inc. (2015), 'Wolfram—alpha personal analytics'.
URL: *http://www.wolframalpha.com/facebook/*