

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 26, 2015

Abstract —

Context/Background Within the past 20 years there has been an explosion in computer mediated interactions between people. There has been much previous work in the area of visualising relationships between 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 will be applied to the representation of dynamic systems with a focus on the interactions between the users rather than simple the static relationships.

Method A system has been created in Python using Pygame to represent various different modes of interactions between users in a virtual environment. 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 representing a wide variety of situations for example the spread of viral advertising, the propagation of to adopt certain behaviours or a model of a trust-based recommendation system. Furthermore it is can be seen that this system of looking at interactions is more effective than currently available approaches in several situations. This is shown by the visual output that my solution has produced which effectively represent the situations listed above in ways clearer than would have previously been possible.

Conclusions In conclusion the results of this solution meet the aims given above. The real-world applications show how my solution is able to represent systems in terms of the interactions between users and how these interactions evolve over time. Additionally they show 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 with other people in virtual environments, whether that takes the form of social networking, email or video games. There is a long history of visualising the structures that form within these environments (Freeman 2000). This has evolved significantly over time with early graphs of relationships being drawn by hand to the present day where layout algorithms running on computers ensures makes it possible to visualise ever more complex graphs with greater numbers of nodes effectively. Finally, recently some of these systems have been made interactive such that they are able to be manipulated in real time allowing increasingly novel ways of visualising data and opening the possibility that these graphs can be used to show more than simply 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 affect other users so that the other users would be able to identify the specific other users they were interacting with. This could take place over a period of time or be instantaneous; it could be between two users or many; it could be a single event or it could be ongoing. For example I will consider interactions such as users sending emails between one another but I will not consider interactions such as a user's advertising preference changing what another user sees as this falls beyond my scope.

Virtual environments in this project will mean any environment in which users are able to interact in the ways I have previously mentioned, mediated by computers. This will most likely mean over the internet, however it could also be the case that it occurs over smaller network such as within an office or university internal network. This could be for example a video game, a social networking website or a messaging system such as email.

A Project Motivation

While there have been many previous attempts to visualise the structure of virtual environments, these have have two drawbacks. Firstly they have only been interested in static 'snapshots' of social networks that either represent their state at a particular time or assume that they do not change significantly over time. This is no surprise as the ability to produce dynamic representations is a relatively new ability in the long history of social network representation. Secondly they have focused only on the relationships between people within these networks rather than exploring other aspects of the system at a more fundamental level.

With increasing ease of interacting in virtual environments and the explosion in 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 the current systems could be extended in two ways.

Firstly I wanted to extend the existing work that covered these static representations of relationships and take what was learned to apply to dynamic systems of interactions that were able to evolve over time.

Secondly I wanted to move the focus of the systems away from the current system of representing only the relationships between users that are inferred from the interactions. I would instead represent the interactions themselves.

B Project Aims

These are the aims for the project, as laid out in the Design Report.

1. Basic Deliverables

- (a) Develop a simple system that models user interactions with a visual output
- (b) Use this system to implement an existing model of user interaction
- (c) Expand this system and model to be able to visualise different modes of interaction

2. Intermediate Deliverables

- (a) Expand the initial system so that its state will evolve over time according to the interactions of its users

- (b) Examine and visualise the way in which users cluster according to their interactions
- (c) Allow the system to set its initial state by real-world social network data

3. Advanced Deliverables

- (a) Visualise change in the clustering of users according to their interactions over time
- (b) Generate quantitative output related to the change in clustering of users over time
- (c) Apply my quantitative output to attempt to solve a real-world problem

II RELATED WORK

A *Social Network Visualisation*

As I have already mentioned, network visualisation is a science with a long history, the interest here however is after computer started being used to assist in the drawing of graphs. There are currently 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 also previously been used to build ways of visualising social network data from Friendster (Heer & Boyd 2005) in order to facilitate discovery of more information than would be apparent from other ways, showing that there is merit in using a graphical approach, and indicating that another approach may be useful.

B *Information Presentation*

There is also a wide variety of information on the presentation of data on computer screens. The most popular model can be summed up as 'Overview, Zoom, Filter' (Shneiderman 1996). In this model it is suggested that the initial view of the data should be 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 zoomed in on preserving the context of the overall picture before extra information of areas of interest can be viewed possibly by clicking on them. This paper also talks about the importance of smooth display updates and responsiveness to user input, something also considered in this project. This is built upon by the ideas of making information more clear by distorting the 'presentation space' (Carpendale & Montagnese 2001). This is the method used in Vizster and can be seen in common usage in many different 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 information more clear. These ideas are also expanded on further to see what kinds of lenses are suitable for which purposes, and mathematical frameworks for implementing such a lens (Leung & Apperley 1994) have been discussed. Together these papers can provide a unified way of presenting graphs in a dynamic way by as the techniques are complimentary. This is the approach that will therefore be taken in this project.

C Graph Drawing

There is also previous literature on the drawing of graphs in aesthetically pleasing ways. Almost all current research makes use of a force directed spring layout. In this algorithm, each node is modelled as repelling each other node and the edges between them are modelled as springs (Fruchterman & Reingold 1991). Included in these papers are suggestions for the strength of the attractive and repulsive forces different distances and the size of graph that this can be expected to create. 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. However, with the correct optimisations it is possible to reduce the complexity to $o(n \log(n))$ (Barnes & Hut 1986). However, 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). It is also possible to use data gathered from a user's social network using tools made for the purpose of downloading such data in a usable form. For example Netvizz (<https://apps.facebook.com/netvizz/>) There are also many models that have been proposed for interactions in other ways. These include models for the spread of influence (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 interactions within 3D virtual environments in which people interact as virtual avatars, referred to as Networked-Virtual Environments. One particular application of this is games (Manninen 2000). Here we can see that there is more than one way of categorising interactions, one way being based on their purpose. These papers also show how it is possible for many different modes of interaction to happen simultaneously. It is possible to use communicative action theory to categorise interactions by their purpose, this is extended in other papers by comparing interactions in a selection of game environments (Becker & Mark 2002). Extending this to other environments such as the social network, other papers show how much of the interaction that goes on within a virtual environment is hidden from the user. We can see just how much data website such as Facebook collect about us including in our making interactions which we wouldn't normally consider meaningful (Schneier 2010)

F Detection of clustering

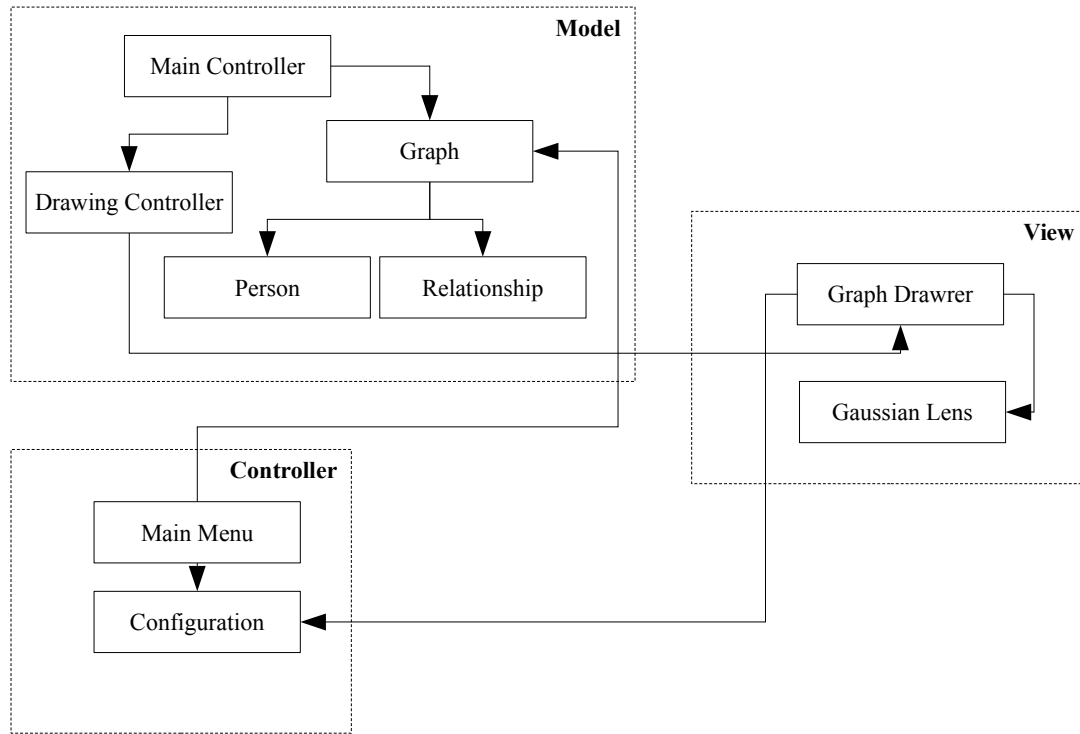
Detecting features of social networks that are not immediately apparent is also extremely important. We can see that algorithms have been developed that aim to detect communities,

related to clustered sections of graph representations of these networks (Newman 2004). These algorithms can be applied to real world networks with a good degree of success reported in identifying the same communities that the users themselves identify with. Given the visual nature of my project I will therefore endeavour to present the graph in such a way that any clustering based on interactions between the users becomes obvious.

III SOLUTION

My solution focuses on flexibility and allows the user to enable and disable as well as configure many different elements. These various elements are described below.

Figure 1: Architectural diagram of solution



As was described in my Design Report, I have used a 'Model, View, Controller' software pattern to guide the architecture of my solution. This has allowed the components to remain as separate as possible as shown in figure 1.

It can be seen in this diagram how the the Model contains all necessary logic in the system along its state with the Drawing Controller and the Graph while keeping this separate from the Controller which handles the Menu generation and storage of its results. The View is then again separate and can focus on its interactions with the Model and the Controller to draw the graph and distort it with the Gaussian Lens.

A Menu System

The primary control of the system is given initially through a system of menus which are presented to the user before the system begins running.

Figure 2: Example menu system of the solution

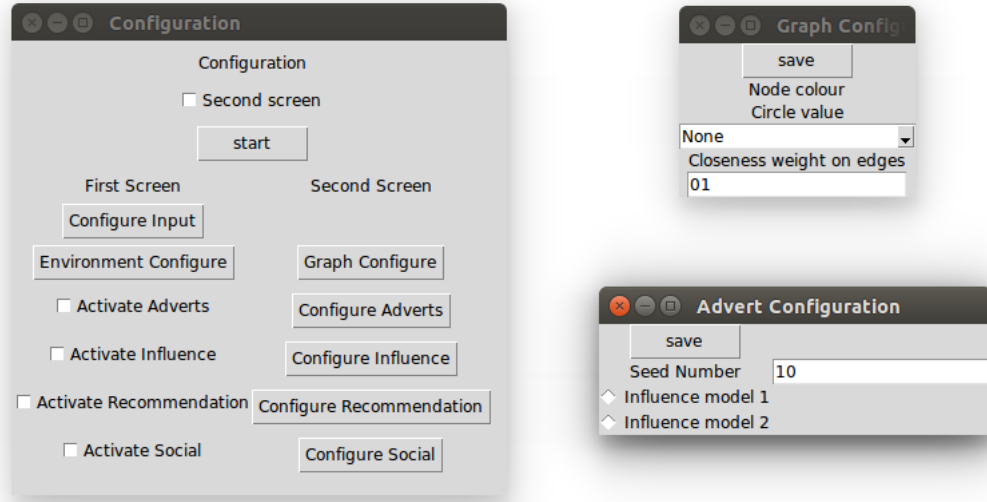


Figure 2 shows an example of these menus. The main window on the left represents the main control of the application. In this window it is possible to enable and disable various models that can be used in order to demonstrate the visualisation system. The buttons next to these check boxes allow various aspects of each the demonstration to be configured so as to give different results. The aspects that can be configured are detailed in the descriptions of each system below.

This menu also allows the user to select what initial input data will be used. They can either generate random data that will be realistic enough in order to demonstrate the system or import their own data. 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, the sex of the users, how many things they like and a separate list of all friendships between the listed users in order to build up a network.

B Interactions with Vertices

The user is able to interact with vertices representing people in the graph in two different ways. These are shown in 3.

The first way is that if the user has selected it as an option in the first menu, 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 to show what information the user requests, in this case it shows the name and sex of the person represented by the vertex.

Figure 3: Diagram showing labels and selctions



The second way in which the user can interact directly with vertices is by clicking on them. This has a different meaning depending on the context for example, it could be used to select the points being used to seed in the advert or influence models laid out below, the colour change is then used to indicate which vertices are selected.

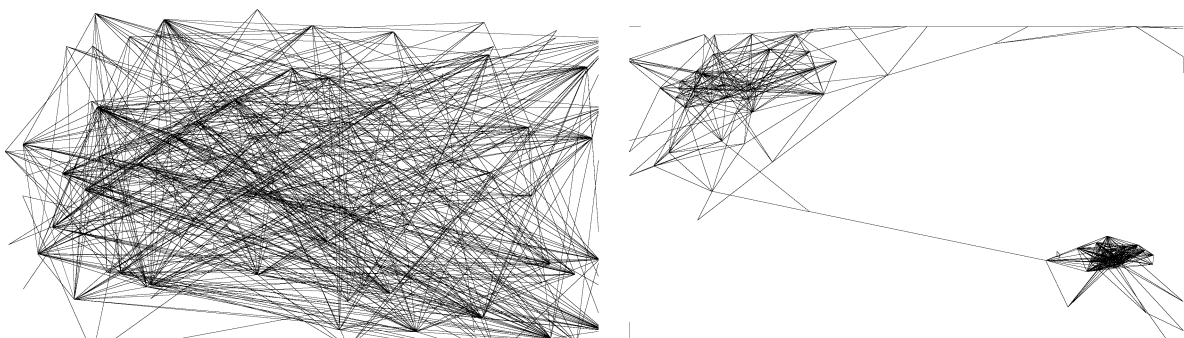
C Graph Layout

This element of my solution is always enabled and allows the system to lay out the graph which forms the basis for for its representation of interactions. The system that I have used to lay out my graph is based on a previous sprint-directed layout algorithm (Fruchterman & Reingold 1991).

Current data visualisations make almost exclusive use of a force-directed spring layout for graph drawing. This method models a graph as a set of springs along each edge joined at each node. These springs, as in my case, need not respond to force in the same way as a real, physical spring but can instead have whatever response gives the best layout for the graph.

The algorithm can then use an iterative approach in order to find a local point of least tension on the springs as a collection.

Figure 4: Graph Layout



I have detailed the algorithm that I used in 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 to be the best.

Ever vertex starts with 0 displacement. The first part of the algorithm then calculates the repulsion between every pair of vertices and updates its displacement an amount proportional

Algorithm 1 Graph sprint-directed 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  $\leftarrow \text{distance}(\text{Vertex}, \text{Other})$  ▷ distance gets Euclidean distance
    diff  $\leftarrow (\text{Vertex}.x - \text{Other}.x, \text{Vertex}.y - \text{Other}.y)$ 
    Vertex.displacement  $\leftarrow \text{Vertex}.displacement + \text{diff} \times (k^2/\text{dist})$ 
  end for
end for
for all Edge in E do ▷ Edge between two vertices, V1 and V2
  dist  $\leftarrow \text{distance}(V_1, V_2)$ 
  diff  $\leftarrow (V_1.x - V_2.x, V_1.y - V_2.y)$ 
  V0.displacement  $\leftarrow V_0.displacement - \text{diff} \times (\text{dist}^2/2k)$ 
  V1.displacement  $\leftarrow V_1.displacement + \text{diff} \times (\text{dist}^2/2k)$ 
end for
for all Vertex in V do
  (Vertex.x, Vertex.y)  $\leftarrow$  (Vertex.x, Vertex.y) +
  (Vertex.displacement/distance(Vertex.displacement)) ×
  (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
end for
```

to the inverse of the distance. The second part of the algorithm then calculates the attraction along every edge and updates the displacement of each vertices in this edge proportionally 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 4. It can be seen how the points are taken from a random layout to a layout in which it is easy to see the structure of the graph.

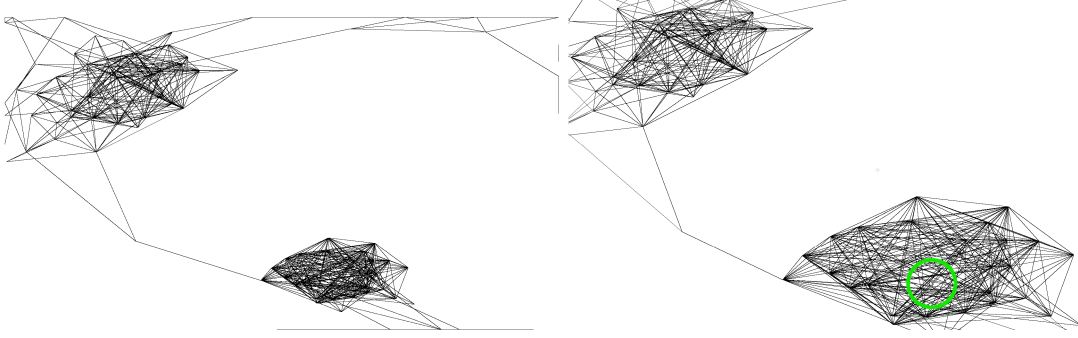
D Lens

Many of smaller details of graphs, especially when they include a large number of nodes can become hidden. After experimentation it was decided that a Gaussian lens would be best suited for this. This allows a gentle falloff and smooth transition through the point of maximum magnification that made the interaction natural as the user moved the 'lens' around the visualisation. Gauss's equation is given by

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

In my solution $f(x, \mu, \sigma)$ represents the distance that the point will be displaced away from the cursor. In my implementation $x - \mu$ 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. I settled on 0.1, meaning that vertices over 20% of the width or height of the screen away would no longer be significantly affected.

Figure 5: Example of lens



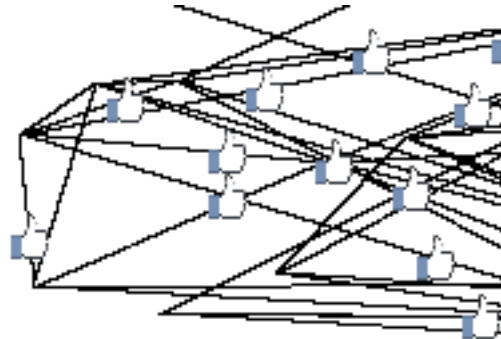
It can be seen in figure 5 that the lens allows the interactions surrounding the cursor (highlighted by a green ring) to be viewed much more effectively.

E Transitions

One way of showing interactions between users in real times is by showing movement in the graph. An example of this can be seen in figure 6. In this example these might be used to show 'likes'. As a user likes another's post a Facebook '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.

It is possible to use this as well to represent other interactions such as the spread of influence. The 'thumb' can be replaced by another image or by a simple disc or other relevant symbol.

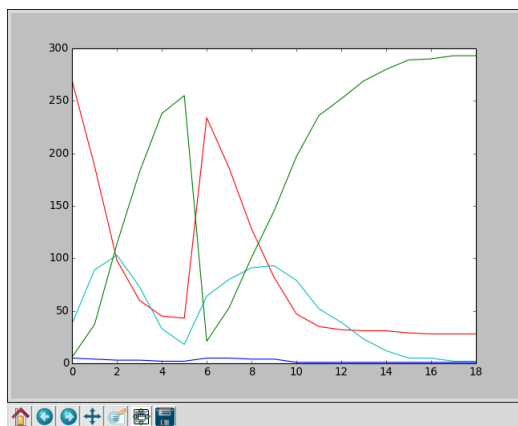
Figure 6: Examples of transitions



F History

In order to make a fast moving interaction more clear, 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.

Figure 7: Example if a display of the history of states



An example of this can be seen in figure 7. 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 graph view.

This visualisation makes it clear how the distribution of users changed over time. In this view the progress of the first campaign can clearly be seen, as can the jump when this campaign is ended and a second introduced.

G Vertices

The system allows for many different ways of representing people as vertices depending on their attributes in order to make it as clear as possible what is being represented.

Figure 8: Examples of possible representations of vertices

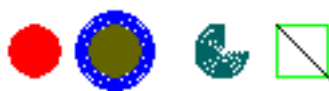


Figure 8 shows examples of different ways that attributes of the vertex can be changed in order to show different information.

The vertex on the left represents the most basic at only one solid colour, this colour could change for example to represent changes of state 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. This could be used to represent multiple properties win the same vertex.

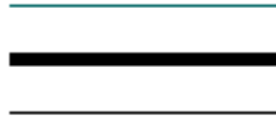
The second vertex from the right shows how it is then further possible to represent properties by the length of a filled arc around a vertex. This could be used to represent the state of a property that it is known will fall between two bounds for example age.

The final example on the far right 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.

H Edges

There are multiple ways of using edges of the graphs to represent the relationships between people. A selection of these can be seen in figure 9.

Figure 9: Examples of possible representations of edges



The bottom line shows how edges are generally represented as a single pixel line joining the two users who have a relationship between them.

The middle line then shows that the thickness of the line can be changed to represent different things about the relationship between two people. This could for example be used to represent the strength of the relationship between two people.

On the right I have shown how it is possible to change the colour of the line joining the two vertices. This could be changed to represent some interaction between the users and can obviously be combined with either of the two previous methods.

Finally it is also possible to add significance to the weight of the lines by adjusting the graph layout algorithm. This could then be used to affect the shape of the graph so that, for example, users who interacted more frequently were more clustered.

I Demonstration Models

In my implementation I included a set of demonstrations in order to show off the usefulness and effectiveness of my implementation as applied to real-world scenarios.

I.1 Influence

I implemented the two models of influence spreading through a social network according to (Kempe et al. 2003). This could be used to model behaviour adoption across a social network, for example if a new social game such as 'Farmville' were to be released, this could simulate how social network users would encourage their friends to also play the game.

In the first model of influence, based on 'Linear Threshold Model', initially each vertex is given an adopted value of *False* a value representing the influence that they are currently feeling and an adoption value, representing how much influence must be exerted over them in order for them to adopt the behaviour. A given set of starting nodes are then chosen and have their adopted value set to *True*. The model then proceeds in steps. At each step, first the weights of all connections to that vertex are added up, then the weights of each vertex that has adopted the

behaviour. If the weights of vertices that has adopted divided by the weights of all neighbour is greater than the vertices adoption value, that vertex then adopts the behaviour.

In the second model of influence based on the 'Independent Cascade Model'. Initially, similarly to before each vertex is given a property to show that it hasn't adopted the behaviour, and a random set are chosen to have adopted it. As before the model then proceeds in steps. In each step, 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 both on the 'strength' of the edge between the two vertices and a random element. Each node that is successfully influenced has only one chance to influence its neighbours, but one node will have multiple chances to be influenced if it has multiple neighbours who are influenced.

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.

I.2 Advertising

A model of the spread of viral advertising was implemented based on a pre-existing method (Van der Lans et al. 2010) which relies on a transition of each person between a number of states. These states are: People who haven't participated in the campaign; people who have received a 'seed' email; people who have seen a traditional advert as part of the campaign; people who have received an email from a friend about the campaign; people who have chosen not to participate and finally people who have participated.

In the initial set-up, all vertices have a status set that they have not participated in the campaign. A random number of 'seed' emails are then sent to people, then a number of 'seed' adverts are shown to the vertices, both with a chance to trigger participation in the campaign. As before this campaign proceeds in steps. At each step, a certain number of vertices being considered will check their email. If they have received an email related to the campaign, either from another user or a 'seed' email then this will give them either a chance to participate in the campaign or choose not to. If they choose to participate in the campaign, they will then generate emails to a random number of their friends about the campaign and the cycle continues.

In this case it is possible to change all of the relevant variables in order to see their effects 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 or even to give different response rates for emails from their friends and seed emails.

I.3 Recommendation

I implemented a model such as might be found in a social recommendation system as inspired by (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, ratings are a value between 0 and 1. 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 opinion of the product, 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 somebody it then takes this recommendation with a degradation depending on how many edges separate the vertex under consideration and the one doing the recommending.

It is possible in this model to change several variables within the system 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.

J Tools Used

The solution is entirely written using a combination of Python 3.4 and various libraries. I used a 64 bit binary of version 3.4.2 of CPython as my interpreter, downloaded from www.python.org. This was chosen as at the time of writing 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 of my computer and python 3 was chosen rather than python 2 so that I was able to make use of recent performance optimisations.

Visualisations were produced using the PyGame library. I obtained this by building the source available from <https://bitbucket.org/pygame/pygame> on my system at the time of writing with CPython as mentioned above. PyGame was chosen because of its ease of use over OpenGL allowing for 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 in the solution 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 easy drawing of simple graphs and would easily allow me to change the style in order to experiment with different ways of displaying the data.

K 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 in order to achieve the objectives set out at the beginning of the project.

Software validation was made in reference to the objectives and functional requirements set out in my Design Report which were designed to allow me to meet the objectives of the project in several layered steps. This was also helped 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.

L 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 done at all stages of implementation in order to check progress against aims and ensure that subsequent work would be able to function on top of old 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 user output but with hopes of testing cases that would challenge the internal models driving the output.

While the system is highly modular, the modules all rely on one another in order to provide a useful output. Given that each module was fairly simple testing focused on an integration and system level enabling a focus to be given to the output visible to the user as was the focus of the project.

M 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 so important to the project.

I attempted to rectify this by using an 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.

Additionally, part way through 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 I made several mock ups to show what it could be capable of. I present those here.

A Viral System

Here I implemented the viral advertising model specified above. I will then use this to demonstrate various aspects of the representation possible in my system.

In figure 10a we can see a representation of the state of all the actors involved 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 clearly be seen to be clustered around the people who have already taken part in the campaign (in green).

Figure 10b clearly shows the kind of configuration that is possible when adding a new campaign during the running of the model along with a set of preset suggested parameters.

Figure 10c demonstrates how the time evolution of the model can then be shown 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

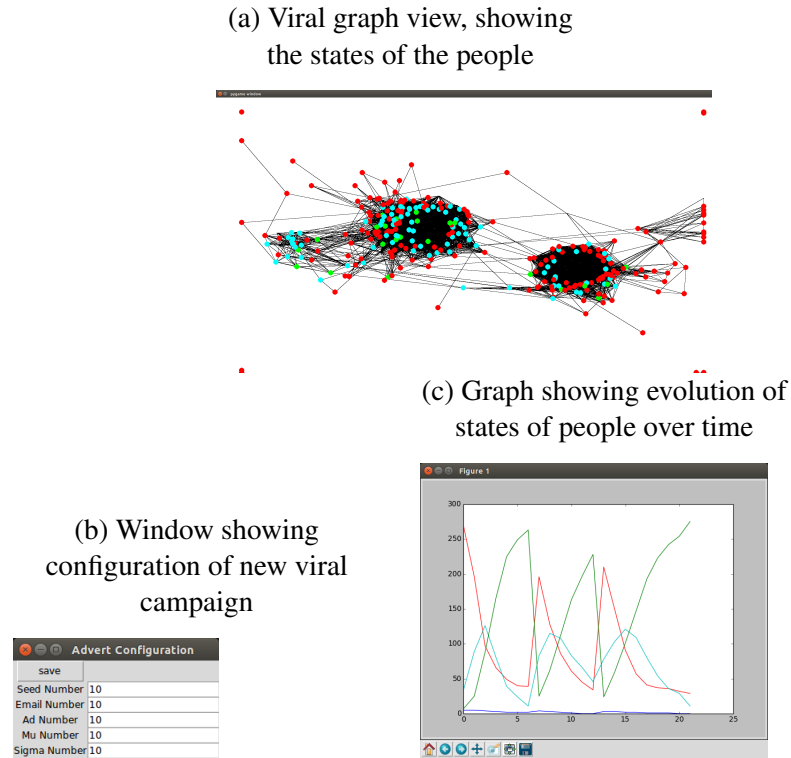


Figure 10: Examples of output produced from Viral model

were used to represent the states on the graph. This clearly allows 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 people who haven't yet participated to outnumber those who have.

B Recommendation System

The same can be seen with the recommendation system which demonstrates a different set of features.

Here we can see in figure 11a 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.

Figure 11b then demonstrates a different way of looking at this data. The colours represent the item which each person is most likely to enjoy.

Then in figure 11c we can see a different layout of the data. Here each of the stretched horizontal plots represents one of the products while the horizontal distance to the right represents how much they are likely to enjoy the product. The comparison between this and figure 11d then shows the interaction between the way different people enjoy different products and the extent to which enjoyment of one product will mean that they enjoy another

C Influence

Finally I did the same with the influence system detailed above.

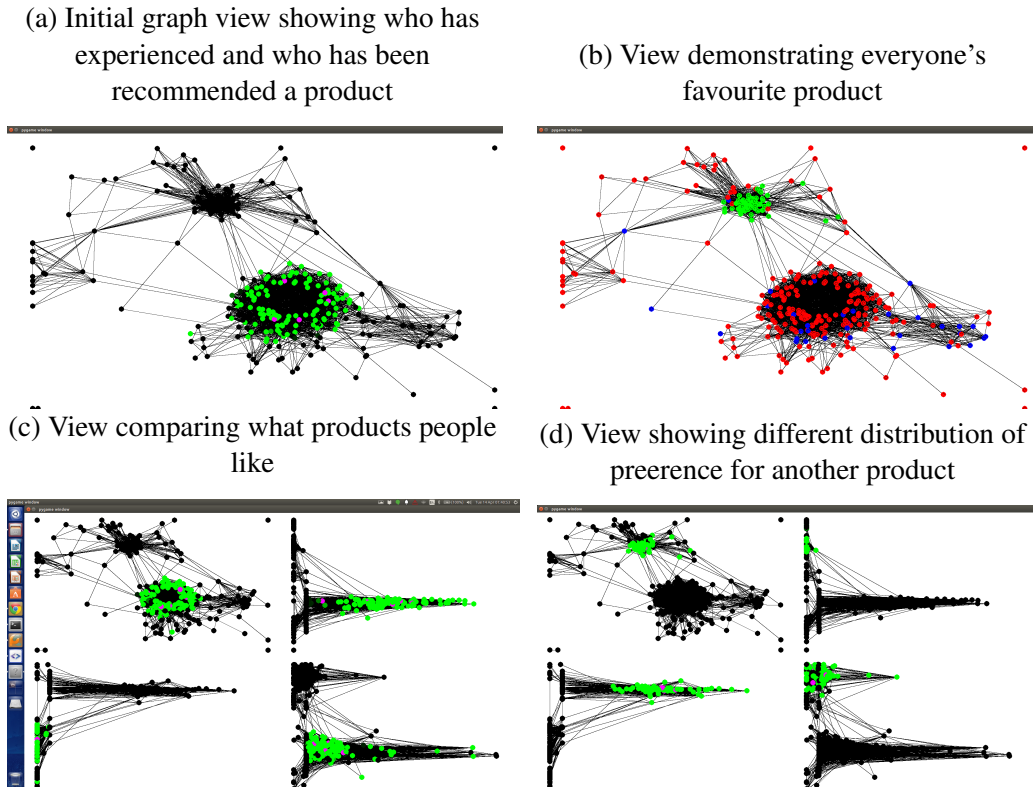


Figure 11: Examples of output produced from the recommendation model

In figure 12a we can see the operation of the user 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 12b 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 differences in the models to be seen in real time.

V EVALUATION

I will evaluate my project 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

I believe it is clear that my system does indeed show the data. This item is included to ensure that actual data is being shown rather than simply results of computation or some other qualitative measure.

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

My solution is to some extent able to allow the viewer to focus only on the substance than the implementation as it is able to

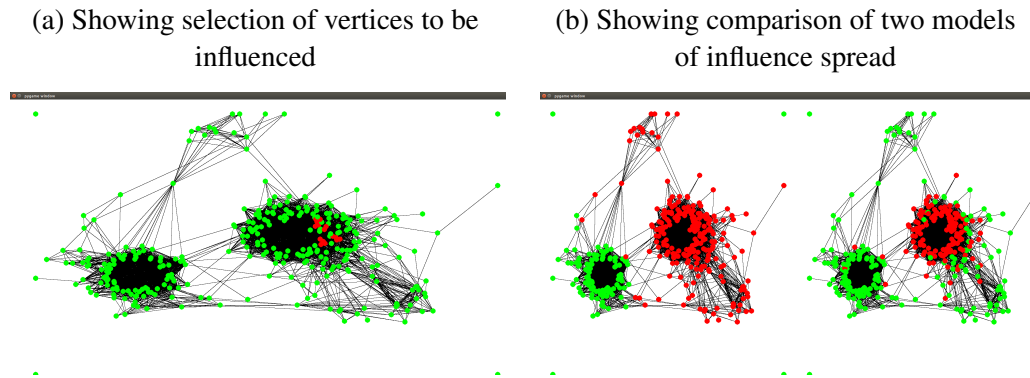


Figure 12: Example outputs from the influence package

3. Avoid distorting what the data has to say

Some aspects of my system ensure that the data is not distorted, for example where a property is represented by something that can have an absolute relationship to a feature of the graph like colour this makes it impossible for the system to distort data. On the other hand in elements like the layout where an iterative algorithm is used to layout with results that are difficult to predict before, this leaves the possibility of distortion if the layout algorithm were to find a poor local minimum.

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. However, my 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, my solution 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 that my system is able to do. 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 12b and 11c. Figure 11c also shows an alternative way of comparison whereby different colours can be applied to the same points in order to show differences.

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

My solution doesn't perform particularly well at this aspect as it follows a broadly flat graph model. However, it integrates 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 this purpose depends on what it is applied to as it is designed that it could be applied to many different data sets. It can be seen for example in the influence application that each element has a clear description, however the exploration of different elements...

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

This is an element that my solution does not address particularly effectively. First of all it was not a part of the project to provide a verbal explanation of the data and in many cases the visualisations themselves have little meaning without such explanation. However, in some cases my visualisations are closely related to statistical representations of the data set.

VI CONCLUSIONS

I believe I have managed to some extent to produce a solution that was able to combine previous research in data presentation of social networks and apply this to a dynamic system of interactions. On top of this I think that my system is able to be applied to various scenarios which show its benefit.

I can demonstrate that I have built on previous research through my implementation of various of its features in the Solution section. This shows that it is indeed possible to use research previously done on static network and apply this to dynamic systems of interactions.

My results show that my system is not only able to extend previous work to include visualisation in new ways that were not commonplace before but that on top of this it is also able to handle similar visualisation to that currently exists simultaneously.

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 each. I hope that these applications are also more useful than previous efforts to visualise them using a static system of relationships and that this allows for greater insight into their evolution.

In the future 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.

References

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

- Becker, B. & Mark, G. (2002), Social conventions in computer-mediated 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.
- 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.
- Newman, M. E. (2004), 'Fast algorithm for detecting community structure in networks', *Physical review E* **69**(6), 066133.
- 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.
- 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.