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 8, 2015

Abstract —

Context/Background

Aims

Method

Results

Conclusions

Keywords — user interaction; virtual environments; visualisation; clustering

I INTRODUCTION

In the 21st century, people spend more time than ever interacting in virtual environments, whether that takes the form of social networking, email or video games. Many previous attempts have been made to visualise the structures that form within these environments (Freeman 2000)

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 of 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 the scope I have layed out.

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 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 almost exclusively focused on static representations of the relationships between users. From this a lot of data has been gathered about information presentation.

As has been seen, much of the previous efforts at visualisation and presentation had focused on the static representation of relationships between people in virtual environments. I wanted to extend this 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

Put aims here

II RELATED WORK

A Social Network Visualisation

Network visualisation is already a science with a long history, especially since being able to use computers to position and draw the output. 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 (**?**). However, these tools have focused on the analysis and visualisation of snapshots of data remaining static, rather than data sets that evolve over time. 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 of looking the data, as I hope to.

B Information Presentation

There is also a wide variety of information on the presentation of data on computer screens. One particularly popular model can be summed up as 'Overview, Zoom, Filter' (Shneiderman 1996). In this 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 things such as the importance of smooth display updates and responsiveness to user input. 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 area also expanded on further to see what kinds of lenses are suitable for which purposes, and suggests a mathematical framework for implementing such a lens (Leung & Apperley 1994). Contained in all of these articles on visualisation are also many suggestions for evaluation of data presentation on computers for example by the ability to maintain context between switching between the three areas on the 'Overview, Zoom, Filter' model and the responsiveness to user input that is possible.

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 is 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 $o(nlog(n))$ (Barnes & Hut 1986).

D Modelling Social Networks

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). Here other relevant ideas are explored such as the privacy implications of collecting data on a large scale and the ability to reconstruct the whole graph from only partial data. The same has also been achieved using the transcript of an internet relay chat (Mutton 2004) again struggling with the problem of reconstructing a complete graph from partial data. It is then further shown that the same method including the temporal decay of relationships can be applied to other sources of relationship information involving over time such as the plays of William Shakespeare.

E Categorisation of Interactions

Previous reserach has also explored categorisation of interactions by their characteristics. This has mainly in the past been applied to social iterations writing 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 writhing 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 Evolution of social networks

Ideas of the behaviour of users in social networks have been the subject of many different papers. This includes homophily (Adamic et al. 2003) which is the idea that people on social networks tend to associate with people who are similar to themselves in terms of age, political views etc. Work has also been completed on the behaviours of users within a social network and the ways in which interactions can spread behaviour across networks of people represented as graphs. It has been suggested that this can be explained using a virus like model (Centola 2010) in which users pass between susceptible, infected and recovered states, analogous to a computer-virus or a real virus.

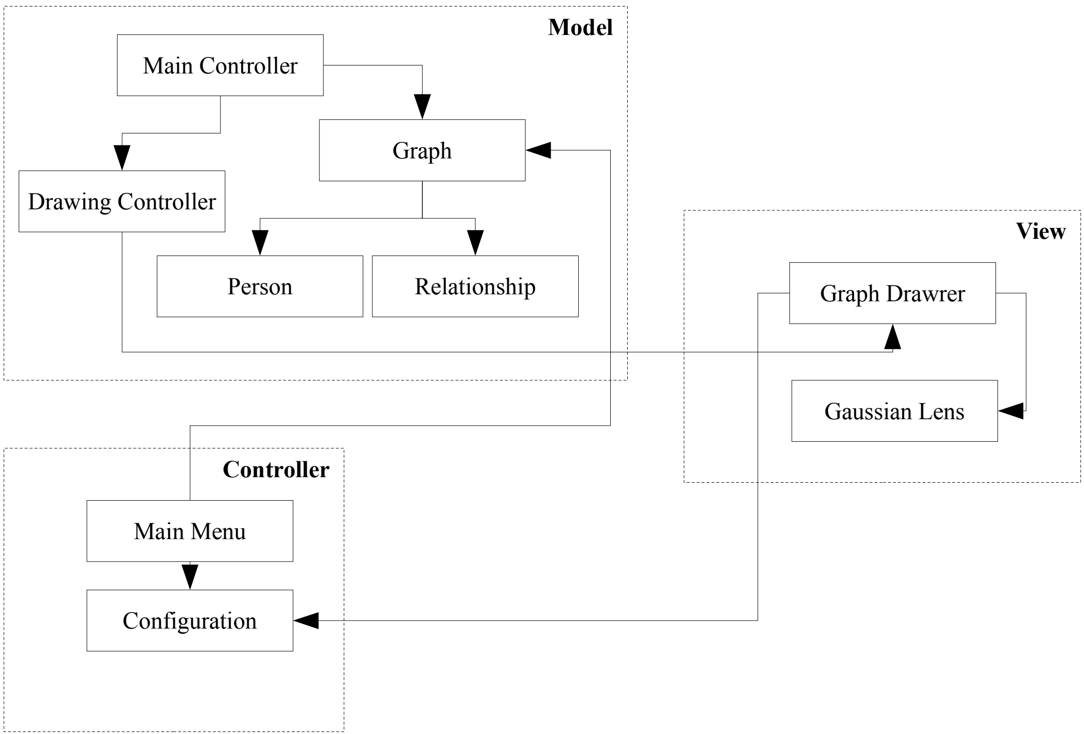
G Detection of clustering

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.

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

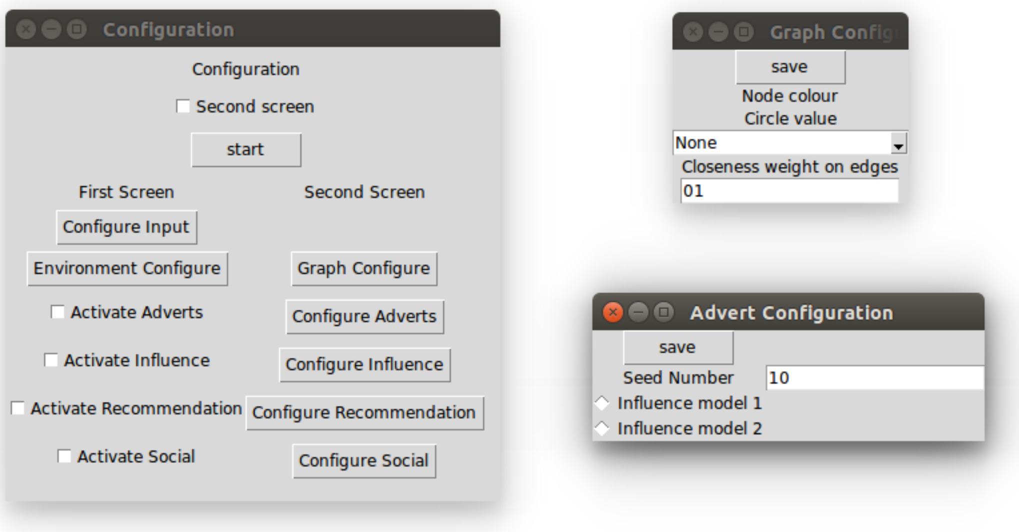


The implementation roughly followed a model, view, controller design pattern in order to ensure that the elements remained as separate as possible

A Menu System

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

Figure 2: Example menu system of the solution



z

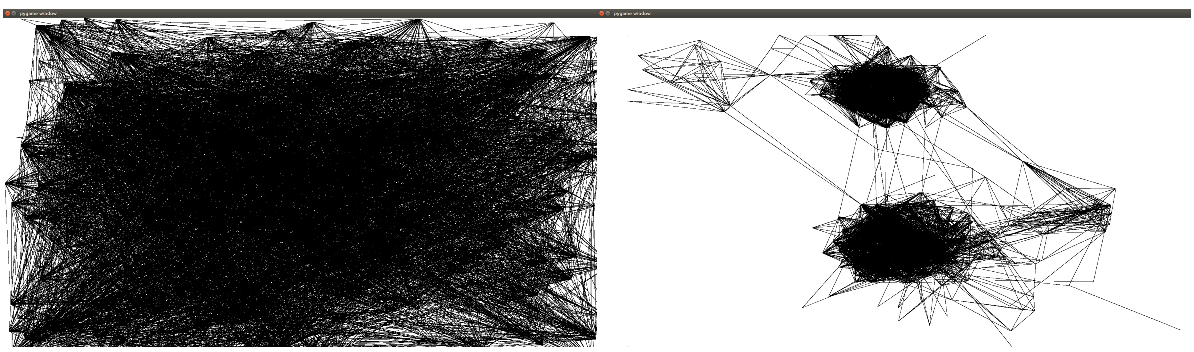
B Graph Layout

This element of my solution is always enabled and allows the system to draw its main view which is a representation of the relationships between the people interacting at that moment. The system that I have used to lay out my graph has been taken form (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 3: Graph Layout



I have detailed the algorithm that I used in Algorithm 1. This is split into three main parts

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**
    $dist \leftarrow distance(Vertex, Other)$ ▷ $distance$ gets Euclidean distance
    $diff \leftarrow (Vertex.x - Other.x, Vertex.y - Other.y)$
    $Vertex.displacement \leftarrow Vertex.displacement + diff \times (k^2/dist)$
  end for
end for
for all Edge in E **do** ▷ Edge between two vertices, V_1 and V_2
  $dist \leftarrow distance(V_1, V_2)$
  $diff \leftarrow (V_1.x - V_2.x, V_1.y - V_2.y)$
  $V_0.displacement \leftarrow V_0.displacement - diff \times (dist^2/2k)$
  $V_1.displacement \leftarrow V_1.displacement + diff \times (dist^2/2k)$
end for
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
end for

after the set up. Initially 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 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 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 3. 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.

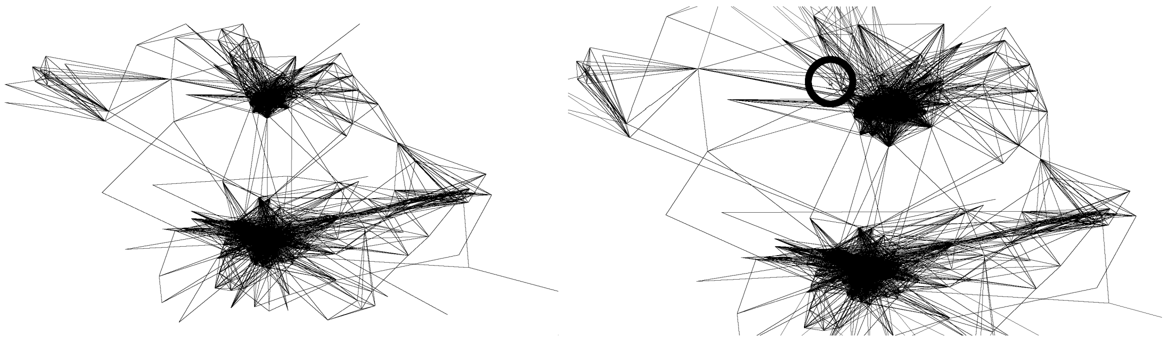
C Lens

Many of these layouts, especially since they include a large number of points in a small area can become hidden. After experimentation it was decided that a Gaussian lens would be best suited for this. This was mainly because of the gentle falloff and smooth transition through the point of maximum focus that made the interaction natural as the user moved the 'lens' around the visualisation.

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

A Gaussian distribution is given by equation 1. In my solution $f(x, \mu, \sigma)$ represents the distance that the point will be displaced away form 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 4: Example of lens



It can be seen in figure 4 that the lens allows the interactions surrounding the cursor (highlighted by a black ring) to be viewed much more effectively.

D 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.

D.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.

D.2 Advertising

I implemented a model of the spread of viral advertising within an online environment. This model was based on (Van der Lans et al. 2010) and 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 setup, 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, again 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 not to, nothing happens, 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.

D.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 on their own

D.4 Social

Finally I took a different approach and implemented a system of interactions such as might happen over a social network. For this I primarily used my personal experiences on Facebook, a platform that I use to interact with people online daily. It is difficult to find reliable statics to use in such a model but in the end I settled on a website which had correlated statistics from many sources of varying reliability (Cash 2015). I was satisfied with this however as it was not necessary for my model to be perfect, only sufficiently accurate to show how my visualisation could be applied to a real situation.

The model encompasses various elements of the Facebook website. Each user has a wall on which they can post either a status of a picture at random intervals, pictures can have one or more friends tagged in them. Each user also has an attribute 'views'. Users will at random intervals check their news feed which is made up of posts from the walls of their friends. Their attitude towards their friends can change according to their attitude towards any other people that might appear tagged in pictures, or the difference in 'views' between a person reading a status and the person writing it.

D.5 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 if 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.

E 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.

F Testing

As I prototyped my implementation, I undertook both static and dynamic testing to ensure that my project was both valid and verified

IV RESULTS

V EVALUATION

VI CONCLUSIONS

References

Adamic, L., Buyukkokten, O. & Adar, E. (2003), 'A social network caught in the web', *First Monday* **8**(6).

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.

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.

Cash, C. (2015), 'By the numbers: 200+ amazing facebook user statistics'.
URL: *http://expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats*

Centola, D. (2010), 'The spread of behavior in an online social network experiment', *science* **329**(5996), 1194–1197.

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.

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.