User Interaction Discovery in Virtual Environments

Student Name: L.A. Sutton

Supervisor Name: W. Li

Submitted as part of the degree of BSc Natural Science to the

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

Abstract —

Context/Background In this project I have taken user interaction to mean any process that happens between two users that changes their relationship in some way. Virtual environments I have taken to mean any computer based environment in which these interactions are possible, this includes social networks, VoIP, interactions with avatars in 3D environments and others.

Aims The aim of the project is firstly to produce an effective way of modelling and visualising these interactions, then to use this model and visualisation to discover clustering of these interactions, and the evolution of the model over time with a view to applying the data to draw real-world conclusions.

Method A system for visualising multiple modes of interaction will be implemented in Python and a an underlying model will be developed based on available literature. The system will then be developed so that it can identify clustering within the network according to their interaction before giving a quantification of this clustering that can be seen changing over time.

Proposed Solution An application that will visualise and model user interactions of many different types and allow their visualisation. It will also allow visualisation of the clustering of users according to their interaction and provide a numerical output that can provide meaningful insight into real-world scenarios

Keywords — user interaction; virtual environment

I INTRODUCTION

A Current Work

A.1 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

A.2 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.

A.3 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).

A.4 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.

A.5 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 symeltaniously. 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)

A.6 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.

A.7 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.

II DESIGN

A Functional Requirements

A.1 Requirement 1

I developed my initial model in Python. I chose Python because of the familiarity that I already had with programming in the language and in the hope that the language would allow me to focus on the modelling as I didn't anticipate the model becoming complex enough that this would slow down significantly any simulation.

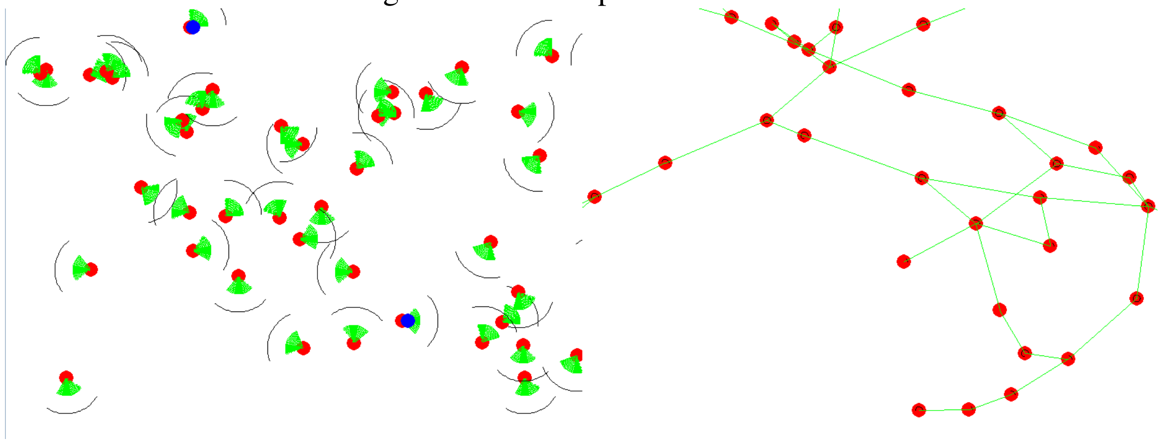
A.2 Requirement 2

The visual output that I used was constructed through the PyGame. I chose this method for several reasons, first I believed that it would allow for faster development and simpler code than producing my graphical output in OpenGL, second the library allows for easy use of multiple CPUs and uses optimised C and Assembly code for core functions, meaning that despite running in Python I hope that the visualisation would still run at a rate suitable for interactivity and smooth animation.

Table 1: Functional Requirements

Requirement Number	Description
1	Implement a basic system for modelling simple user interactions in a virtual environment
2	Implement a visual output for this model
3	Expand the implementation to cover multiple modes of interaction
4	Implement an evolution of the model over time with the ability to quantify the evolution
5	Implement a grouping/clustering of users in the virtual environment based on their interactions
6	Implement visualisation of the clustering of users
7	Implement a way of outputting data that can be shown to have real-world applications

Figure 1: An example visualisation



An example of the current visualisation can be seen in Figure 1. Here on the left we can see a representation of the people in their 3d environment and on the right we see a representation of the relationships between the people .

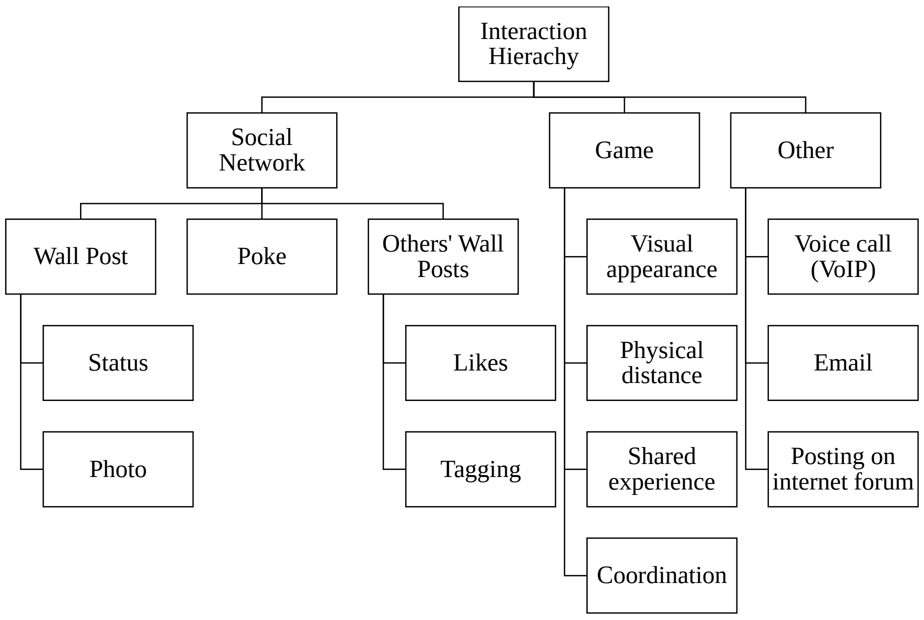
A.3 Requirement 3

When considering multiple modes of interaction I found that there was little current literature linking interaction modes with categorisations. This lead me to then design two hierarchies. The first in order to categorise what different interactions were possible within the virtual environments I was considering, the second to categorise them in terms of how they could be represented.

Initially I produced Figure 2 in order to categorise the types of interactions that were possible. These came largely from personal experience and I produced this in order to help me consider what categories of interactions it would be necessary to visualise.

This lead me to Figure 3, which are the modes of interaction that I will be visualising.

Figure 2: Interaction Types



A.4 Requirement 4

The model will evolve in three ways, firstly the people will move around in their 3D environment. They have random motion around the environment and aren't allowed to move out of its bounds.

The second way that the model will evolve is in the change in relationship strength between the pairs of people. This will be affected by the interactions between the users of the relationships being affected and will also be affected by time as people who don't interact lose relationship strength.

A.5 Requirement 5

A.6 Requirement 6

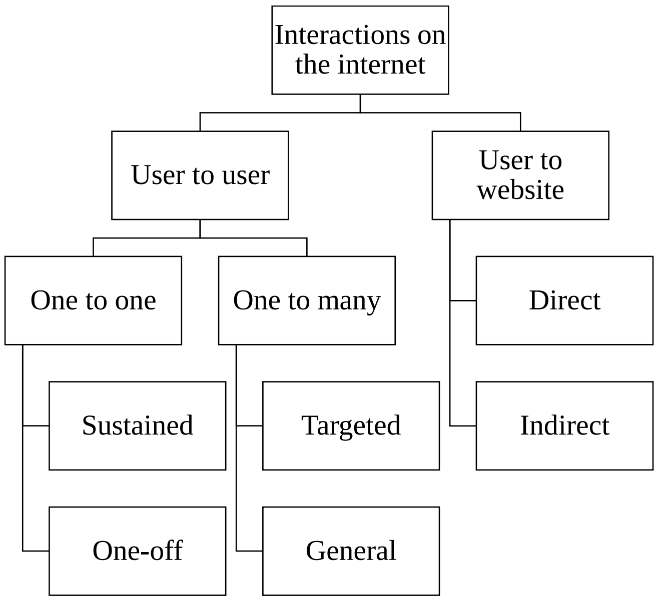
A.7 Requirement 7

B Architecture

B.1 MainController

The MainController class is the one that coordinates the setup and initialisation of the program along with choosing what interactions will be represented. This also coordinates the synchronisation of frames between the two Drawing objects in order to draw them together.

Figure 3: Interaction Modes



B.2 Graph

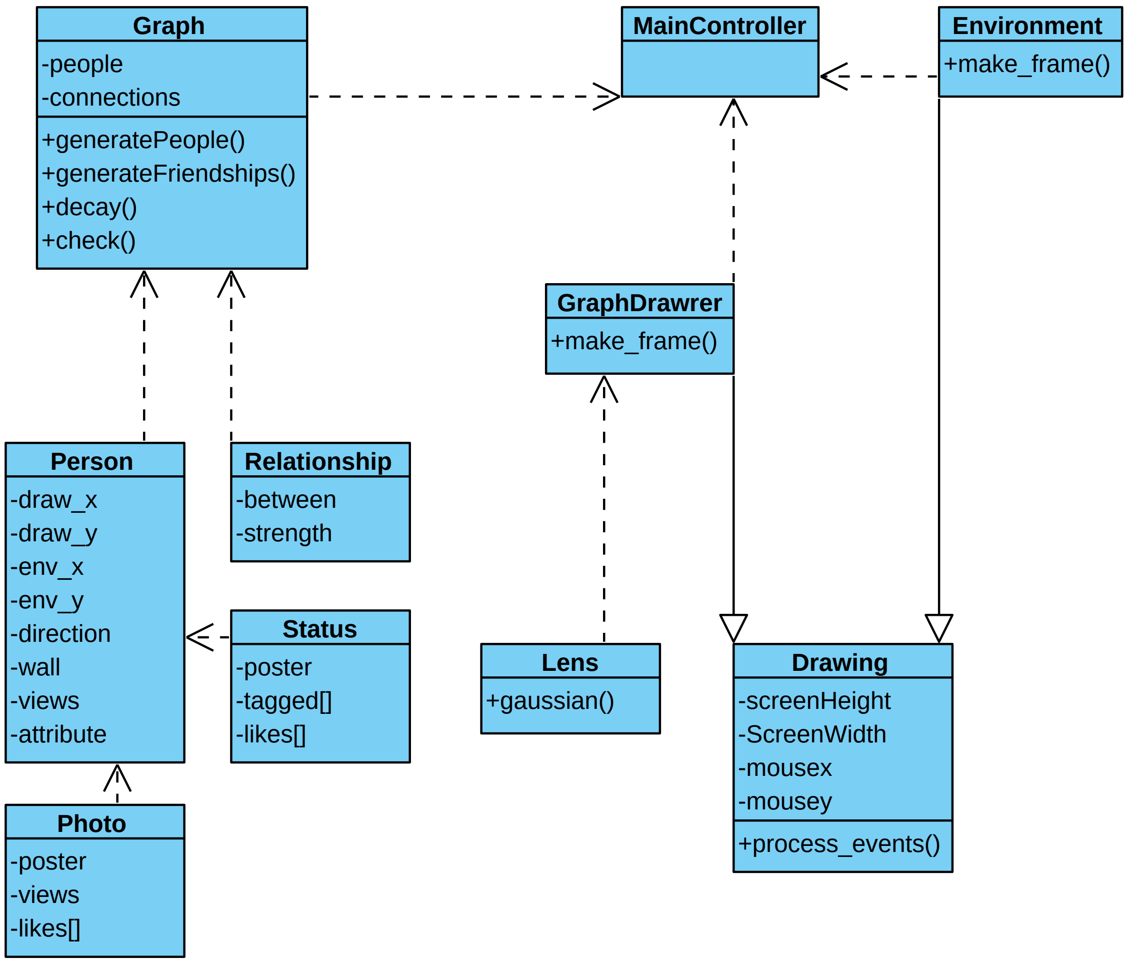
The Graph object contains a list of Person objects (people) and a list of Relationship objects (connections). This stores all of the people that are being considered in the virtual environment and the relationships between them. This object also handles the initial generation of people through the generatePeople() method which randomly generates a predetermined number of people with random attributes. the generateFriendships() method the determines a predetermined number of relationship objects between the people with random strength.

The decay method if called will iterate through the relationships and modify them all by a set amount. This is called in order to simulate the decay of relationships over time between pairs of people who do not interact. The check() method will again iterate through all relationships and check that they are all legal. That two relationships haven't been created between the same pair, it will also remove relationships once they fall below a certain level.

B.3 Person

A Person class represents a person in the virtual environment and forms part of the graph. The person object contains all the attributes of the person including the draw_x and draw_y fields which represent the point at which this person is being represented in the graph view and the env_x and env_y fields at which this person currently is in the 3 dimensional environment. The direction field then applies to the direction in which this person is looking in the 3D environment.

Figure 4: Class Diagram



For the social side of the virtual environment, the person object includes a wall, which functions in the same way as a wall in Facebook with the photos and statuses being modelled on the wall. Finally for the social network each person has a field which I have called views. This represents the persons social or political views that other people might agree/disagree with and are represented in social network statuses

B.4 Relationship

A Relationship object represents the relationship between two Person objects. The between field contains a tuple representing the people the Relatinonship is between and the strength field indicates the strength of the feeling between the two people

B.5 Status

A status is simply an object that can be posted on a wall. It contains as the poster field a reference to the Person object who posted it on the wall. The views field represents the views of the poster which the person who looks at it might either agree or disagree with and finally the likes[] field contains references to the Person objects who have interacted with the status by liking it.

B.6 Photo

Similar to a status the Photo object again represents something posted on a wall. The only difference this time is that it contains in its tagged[] field a list of the people who are tagged in the photo, changing their relationship with the tagger, the poster and the person on who's wall the photo was posted.

B.7 Drawing

The drawing object contains the methods and fields that are common between the two types of drawing in the representation. The constructor fills the screenHeight and screenWidth fields and then every time a drawing calls the process_events() method the object checks if it should close the window and exit of if a key has been pressed changing the representation of and then updates the mousex and mousey fields with the current mouse position

B.8 GraphDrawerer

This inherits from the Drawing object and draws a graph of nodes and edges representing all of the people currently being considered int he virtual environment and the relationships between them. It constructs the layout of this graph by means of a force directed drawing algorithm to place the nodes in the clearest possibly arrangement in a reasonable amount of time. It does this by assuming that there is an attractive force of x^2/k along each of the edges where x is the distance and k is a constant and a repulsive force of k^2/x between each of the pairs of nodes. Several iterations of this are taken as each node is moved by the resultant 'force' on it, changing position less each time until a local minimum has been reached. The make_frame() function then produces the next frame of the visualisation that can be rendered with others.

B.9 Lens

The lens object contains one method associated with the drawing of the graph as distorted by an imaginary lens. It calculates the result of a gaussian function based on the distance between two poiwnts. The function used is

$$f(x) = (1/(\sigma\sqrt{(}2 * \pi))(-(x-b)^2/(2c^2)). \tag{1}$$

This can then be used to separate points for closer inspection.

B.10 Environment

This class is responsible for drawing a representation of the characters in the 3D environment. it takes the env_x and env_y fields from the Person class and uses this and the direction to draw the people at these points. Calling the make_frame() then produces the next frame of the visualisation to be rendered alongside another.

III EVALUATION

IV REFERENCES

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.

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.

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

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.

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.