UNIVERSITY OF OSLO
Department of Informatics

Exploring 3-dimensional visualization of the data sets collected by cfEngine

Master thesis

Alexander Semenov

Spring 2006

Exploring 3-dimentional Visualization of the data sets collected by *cfEngine*.

Alexander Semenov
Department of Informatics
University of Oslo

June 6, 2006

Abstract

cfEngine or Configuration Engine is an administration tool used to configure and maintain computer systems. There is a part of *cfEngine* that is responsible for collection of information about systems' state. As a result, it produces a lot of data that can be shown as a graphs, lots of graphs. Operator can not observe all these graphs simultaneously. We propose a system that generates all these graphs as objects in 3-dimentional World. We will show that the system presenting many graphs simultanously in a new and unusual type of environment can benefit from changing traditional 2-dimentional windowed look to the 3-dimentional Virtual World. Graphs are located in 3-dimentional space according to their importance. File format must be standard compliant as much as possible in order to provide a results that can be viewed in general browser and that appear in different browsers in a similar way.

Contents

Keywords

Presentation of information, *VRML*, *X3D*, scene graph, *cfEngine*.

Structure of the paper

This paper is organized as following. First we will discuss some methods of presentation of information. The next section presents Virtual Reality Modelling Language. Next we will take a look on *cfEngine*. Finally we will present ideas used to design prototype software.

Chapter 1

Introduction

Amount of data we have to deal with is increasing all the time. In some cases traditional methods of presentation can not give a full picture. We will try to find out what kind of improvements can be done to change the way we are working with data. The main idea is to go from traditional 2-dimentional interface that we all know as GUI or Graphical user interface, to the 3-dimentional interface based on Virtual Reality Modeling Language (or simply *VRML*) and principles of Zooming User Interface (ZUI). User should be allowed to explore and interact with the datasets, use point-and-click for user feedback and dynamic coloring and labelling. The questions discussed in this paper are following:

- Can 3-dimentional presentations help us to visualize and monitor large amount of information?
- How information can be structured and introduced with respect to 3-dimentional space?
- Can *VRML* (and new *X3D*) standard be a good choice for data visualization?
- Can 3-dimentional presentations enhance and change standard user interface as we know it today?

We are using open standards and existing systems as much as possible. As an example and a test-bed for ideas we use data collected by *cfEngine*, a Configuration Engine. *cfEngine* is a powerful tool for installation and maintaining of computer systems. It implements principles of "Immune System" concept. Immune systems, as we know them from Nature, protect animals and Human beings from dangerous factors such as bacteria, toxins and viruses. Computer system that is "Immune", will akt the same way as biological organisms do, finding and detecting objects that are not belong to system, isolating them and removing them afterall. To detect malicious intruder, system have to learn about itself. This is important to be able to detect which code is malicious and which one belongs to system itself. This part, including data collection and analysis is done by *cfEngine*. *cfEngine* learns about "normal state" of the system. Information about normal state can be extracted for investigation by a system administrator. During this project we designed a prototype that visualize data from *cfEngine* in an irregular way. Visualization is trying to focuse on the data that system means are important. Data that system detects to be ordinary is also accessable, but transfered to the background. Prototype still can not be recomended for distribution with stable version of *cfEngine* suite. The ideas tested on prototype can be implemented in next improved version of software. The work waas concentrated around concepts of presentation, navigation, interaction and implementation of ideas using standard Internet based modelling language. The objects that system represents are simple charts in our case. We hope that systems administrators that run *cfEngine* and can benefit from better graphical presentation of system state.

Chapter 2

Presentation of Information

A picture is worth a thousand words.

Chinese proverb

Computer systems became more and more powerful. We utilize more CPU power, huge amounts of system memory and space on hard-drives. Our network activity increases all the time. There is a lot of data we have to deal with if we want to be sure that we know what happens in our computer system. We do not know when the event we are interested in can happens. The way of effective working with information is given us in Visual Information-Seeking Mantra, that says, - ”overview first, zoom and filter, then details on demand”[1]. Human brain is a tool designed to images and patterns recognition. Our ability to working with images is much better than working with text. Although text reading is an image recognition process too[2].

A traditional way of collection information about system state is log files. Modern logging systems are flexible and can record almost everything that happens in computer system. Classical logging systems as we know them from UNIX use text files to save logs. Microsoft™Windows™uses it's own internal logging database. Both produce lots of lines with text that are ment to be read by an operator. The amount of data makes this mission absolutely impossible. The power and a weakness of log files is the form of presentation, a text format. Information encoded in a form operator can read and that can be sorted and filtered in a different ways. In the same time, a stream of textual information from log file is hard to understand and analyze. To understand the message, we need to read the message. It takes time. The amount of information is normally huge that doesn't make the task easier. Even psychologically, the volume of work can prevent operator from doing a task that is really important[3]. Analysis is en essential part of preventing intrusions. Intrusion detection system detects if someone is trying to break security of the system. The problem is that fault alarms happen sometimes. Many fault alarms make that operator is missing concentration. Fault alarms can be a reason to miss a real alarm. The speed of reaction is also important. Operator has to be fast enough to detect break-ins. Long detection time can follow to more damage in the system. Pre-intrusion scans cans last over hours or even days. It is great if system can handle this situations using long-time memory. We use of computer-aided, interactive, visual representations of abstract data to improve our understanding of processes inside of the system[4]. Data can be filtered. Applying new filter all the time operator can focuse on particular type of information. Data can be filtered according to the hardware connections between parts of the network, domain policy or different kinds of users activities. The process in similar to zooming on the image to get more details. That's the right way we learn from Visual Information-Seeking Mantra[1]. But information is still needs to be read from the textual form. According to researchers, human eyes are most effective when work with small portions of information. When working with

information, muscles of human eye feel most comfortable moving fokus on a distance about 6 inches. Longer distances make us more tired and reduce our capability to work with information in effective way. That's why text with multiple columnes can be easier to read than the same text fitted into one single column. Ordinary computer systems have screen size about 15, 17, 19 inches. Bigger screen makes possible to show more infromation. But it doesn't increase our capability to work with information. But if we can increase density of information per square inch, then we can provide more information inside of 6 inches limit. One of possible solutions is to combine information to reduce the total amount of data that have to be presented. But, when we combine information, we can loose some details. We can change from standard 2-dimentional presentation to the 3-dimentional[5]SVisionfisk-immersive.

In addition for standard scrolling that we can meet in all 2-dimentional documents, that consist of possibility to scroll up and down and, in some cases, to the right and to the left, 3-dimentional inteface give more freedome to scroll in depth. In the same time it will demand special skills from operator to navigate in this environment in efficient way. Some data such as CPU load, amount of incoming and outgoing traffic, volume of free space on hard drives can be measured and stored in state database and can be extracted later into the form of tables. Data from tables can be visualized using standard charting tools (such as gnuplot, and even Microsoft^{TM}Excel^{TM}). Users ability to understand given information is one of bootlenecks[1]. Data can be viewed in raw form of tables. Tables are much easier to see in form of graphs. When we have many graphs, we need some other methods. How to map information into the graphical form in a most effective and user-friendly way. So our challenge is presentation and navigation through a huge amount of information. Let us find methods that will allow representing information to user in a best possible way. 2-dimentional interface has a single front perspective. Operator has only one point of view. 3-dimentional presentation has lots of perspectives lots of points of view. But this richness has strong connection to navigation. All objects can be viewed from all sides, from top and from bottom. That's why navigation is so important in 3-dimentional world, that's why navigation can be a nontrivial task. From the other side, 3-dimentional scene is still have to be projected onto the 2-dimentional flat screen. It is also important to keep in balance the amount of information presented to the operator. With too much information, it can be difficult to recognize elements[6]

What is important for operator wirking with information?

- recognition of elements
- understanding of how elements of system connected to each other and how changes in different elements can change the system state.
- prediction of the future state based on past and current state of the system.

In some situations when time is important, graphical representation can help to make a right decision as fast as possible. Data from many sources should be integrated into one clear big picture. Well known types of security breaches have patterns that can be recognized with help of visualization techniques. But, well known patterns can be detected by automated Intrusion Detection Systems. Even more important for visuzlization is that new, unknown types of attacks can be resognized and prevented by operator[6]. Let us take a look on how data represented in a traditional way.

2.0.1 Line Charts

Line Charts are used to compare continuous datasets over some interval of time. Line Charts are useful when one need to show how data changes with time. Graph showing the progress can help to predict future trends. It is usual to represent time coordinate on X-axis and function value in a particular period of time along Y-axis. Colors can be used to represent different datasets.

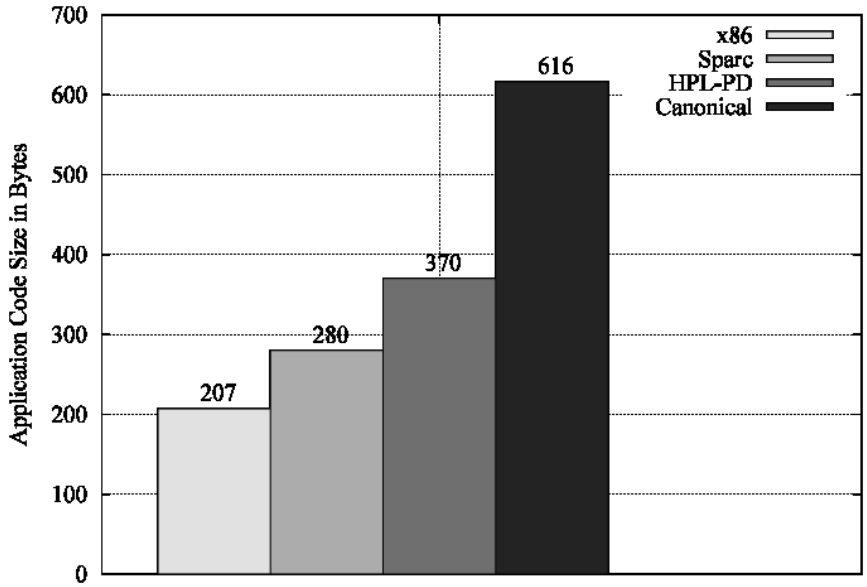

Figure 2.1: Example of Bar Chart diagram created using gnuplot.

2.0.2 Bar Charts

Bar Chart is a diagram created of rectangular bars. Bars can be oriented in both vertical and horizontal manner. The height or length of each bar represents the frequency. In some cases when there is no reason to start diagram from zero level, then bars are not proportional to values but proportional to differences between values. A type of diagram that is useful to show differences between datasets. Some bar charts can be designed in semi-3-dimentional way when each rectangular is transformed into the 3-dimentional bar with constant depth. Another type of diagram similar to the Bar Chart is Histogram. Using histogram is a good way to describe distributions of continuous variables. The area of the graph represents frequency and the height of histogram area represents frequency density.

Pie Charts. Is a way to show relation between a dataset and the whole amount of data. Diagram has a form of pie with sectors (or slices) proportional to percentage of each value from the total amount of data. To focuse on a particular part of the chart, some segments can be partially taken out from the graph. This effect known as exploded pie chart. Radar Chart or Spiderweb Chart. Can be really useful when need to compare several different datasets related to one item. Chart consists of severla axes along which data can be plotted. Points close to the center of the chart have the lowest values.

Multidimentional charts can be used for presenting more complicated data. Matrix chart is a form of table with categories in raws and columnes and with each cell containing a bar. Bars length and colour add extra dimention into diagram. Matrix diagram can be presented in 2-dimentional space, as a flat table. With adding 3-rd dimention Matrix chart will be changed into Cityscape chart, a form of 3-dimentional Bar Chart. Adding walls around 3-dimentional Cityscape chart can provide place for aditional information[8]brown00network. CityScape diagrams are also known as Stacked BarCharts, a set of BarCharts that are set together.

Combining properties of different charts we can reproduce data in more concentrated form. Star Maps is a chart that can be described as a histogram rotated at a single point. Each value can be drawn at a degree from 0 to 2π. The resulting diagram looks like a star. Different colours can be used in addition to include more information into diagram.

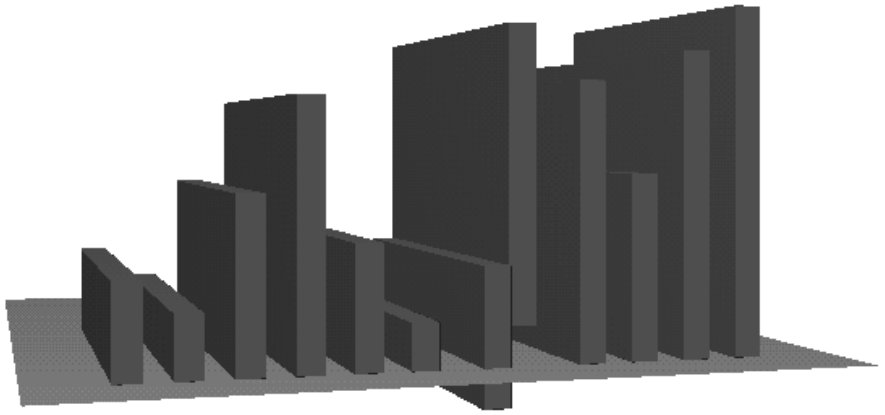

Figure 2.2: Semi-3-dimentional *Bar Chart* representing bubblesort visualization made by Polka-3D system[7].

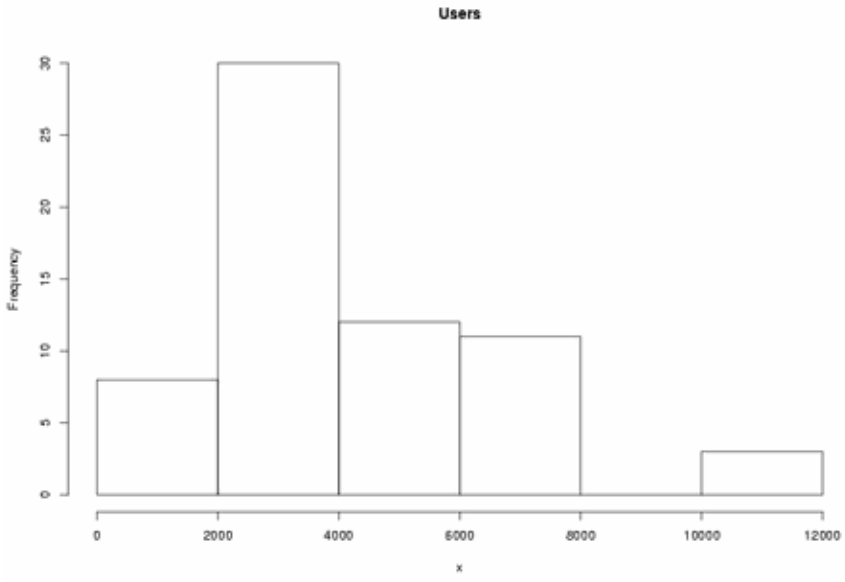

Figure 2.3: Example of Histogram created using gnuplot.

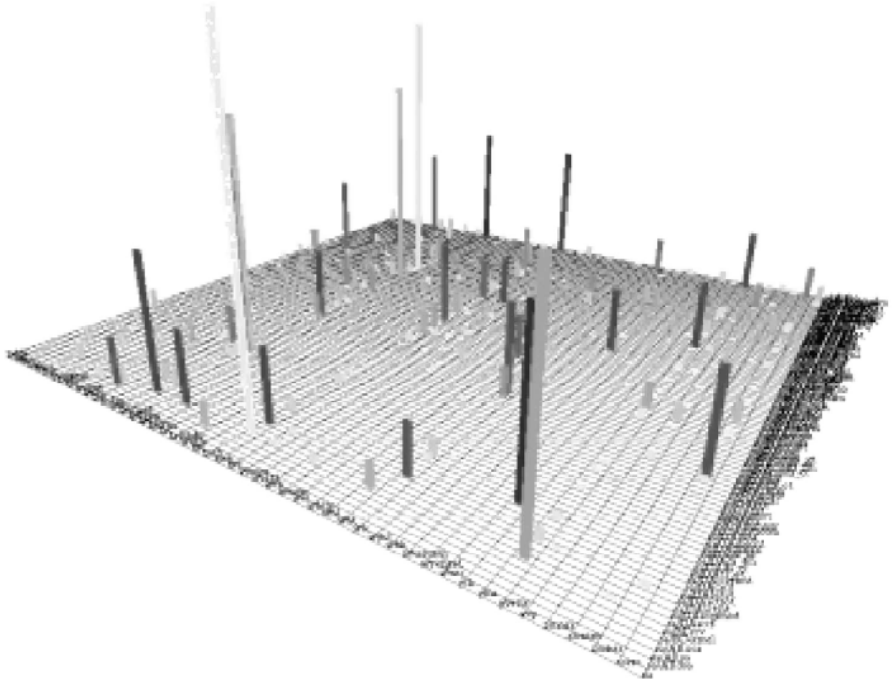

Figure 2.4: Example of Sityscape chart[8].

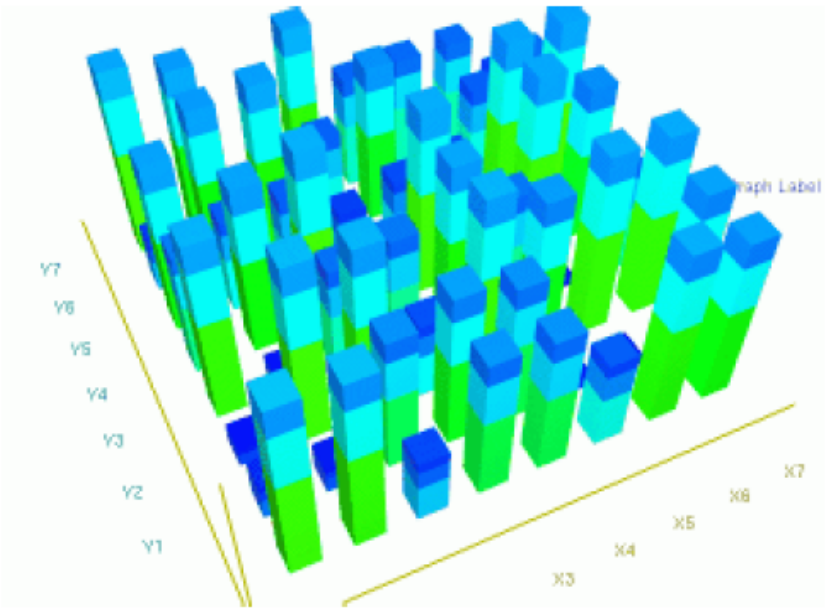

Figure 2.5: Example of Stacked BarChart created by Cichlid[9].

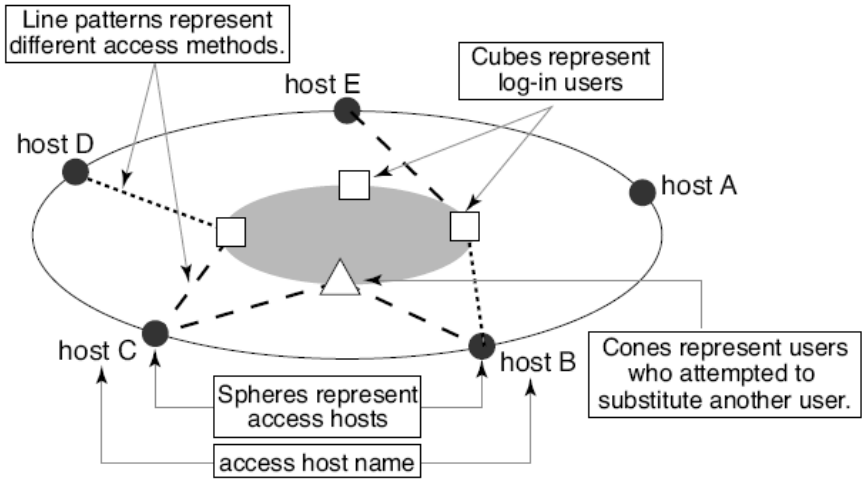

Figure 2.6: A simple unit of visualization created by Tudumi[3].

Star diagrams can be combilned into tables or matrices. Location (or placement) inside of matrix gives us additional level of information. Star Map diagrams can even be stacked together giving 3-dimentional picture[1].

Complicated systems demand that operator will use some time in the beginning to be more familiar with a way they reperesent data.

2.1 Examples of existing systems

Let us take a look on some approaches to present information

2.1.1 Tudumi

One of the approaches to visualize information is a program named Tudumi. It's a system that can visualize computer log files. Tudumi changes textual based environment of log files to the environment based on images. Log messages are sorted and summarized on the preparation stage before applying rules and visualizing results. Rules can be altered dynamicaly making immediate effect on visual representation. A prototype of Tudumi concentrates on particular types of activities connected to users. Tudumi uses sets of concentric disks. The lowest disk is dedicated to represent user substitution information, which is a critical task, and in many cases can be a sign of intrusion. Other disks dedicated to users login and network access information. Tudumi depicts hosts (represented as spheres on the outer border of concentric disks) and logged users (represented as cubes on inner circles) and relations between them. To distinct users from each other colors or textured images are used. Some users change their pictogram from textured cube to red cone representing that user substitution activity was detected. Users related by a substitution are mirrored to the bottom circle and connected to an actual login pictogram with arrows. Different patterns in lines connecting hosts and users can represent different kinds of activities such as terminal activity or file transfer. Activities are sorted according to rules and what we can call a normal activity will sink to the bottom. Suspicious activities will follow up the graph until the top plate where the most suspecious cases are. Grouping similar events and using interactive filtering Tudumi can zoom on data that is significant.

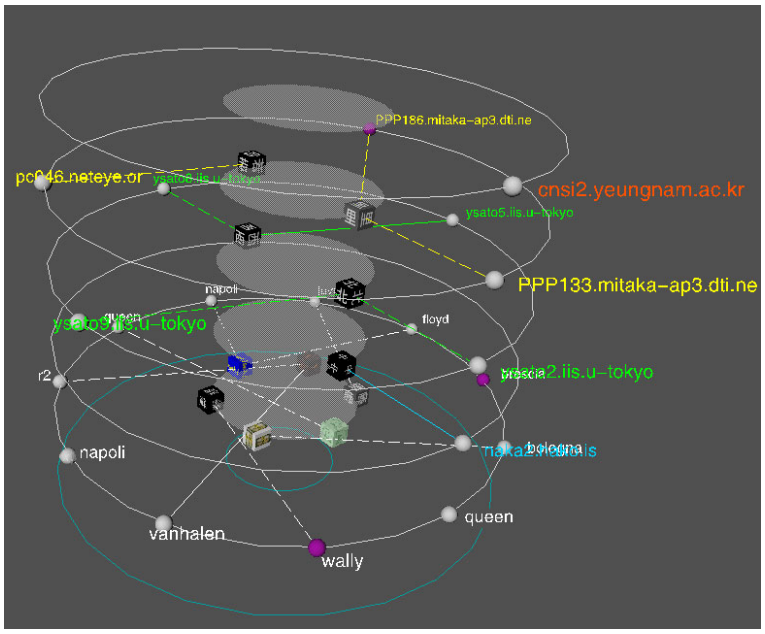

Figure 2.7: Visualization example from Tudumi[3].

2.1.2 Mielog

Another interesting approach is represented in a Highly Interactive Visual Log Browser or Mielog[10]. It uses statistical analysis to separate unusual events from events of normal behaviour. The screen in Mielog has 4 areas, 3 for visualization and the last one for displaying actual log messages. To make analysis easier, system converts and compiles conventional log files into one file of "Generalized Log Format". The Tag area of visualization represents color code according to frequency of message ocurence. Area uses colors of gradient from blue to red with blue representing normal behaviour. Next area is a Time area. It is divided into 3 subareas that represent frequency information respectively to day of the week (the 1-st subarea) and time of the day (the 2-nd subarea). The coloring is similar for the Tag area, but colors are taken from gray scale. The last subarea contains a form of histogram with bars showing amount of messages located in that particular area. Outline area contains scaled bar chart where length of bars is proportional to the lenght of log messages and colors of bars are according to the same rules as in Tag area. Message area contains actual log messages in original text form. As a result, operator do not need to read through log messages, but can recognize image patterns. Color makes easy to see if message is a sign of unusual behaviour. Extra highlighting of words and phrases is used in the Message area. Mielog includes a rich set of interactive filtering options that can be applied in all areas[10].

2.1.3 Cichlid

Another very intersesting example is a system named Cichlid[1]. Cichlid is a tool for visualization and analysis of computer networks. It is written in C and is using OpenGL and GLUT and produce dynamical 3-dimentional images of collected data. Cichlid is a distributed system, information collected on various hosts in the network (called DataServers). Then it is transmitted to the visualization client using TCP. Cichlid implements various function such as zoom, point to focuse and move/explore. Methods of visualization include

[1] created at National Laboratory of Applied Network Research (NLANR), San Diego Supercomputer Center, University of California

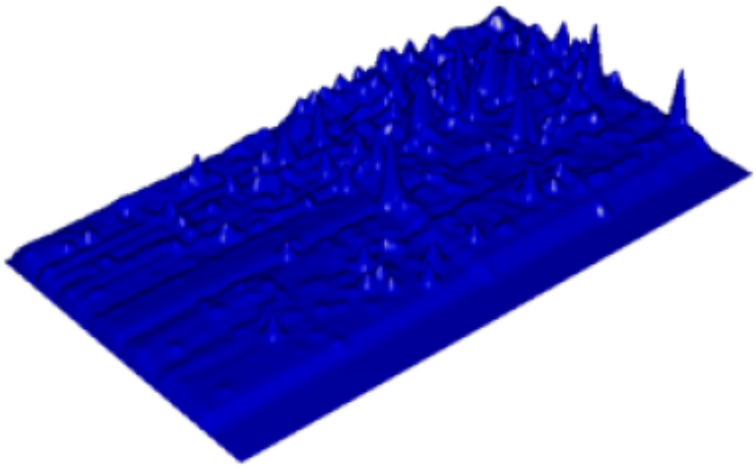

Figure 2.8: An example of network "Terrain" created by Cichlid[9].

Stacked BarCharts (that can be transformed into interpolated surfaces) and vertex/edge graphs representing network topology[9]. Cichlid is a cross-platform application and can be run under FreeBSD, Linux, Microsoft Windows, and IRIX platforms. Design of Cichlid allows users to create their own modules that will transmit data collected in a way predefined by user. To make data transmition more effective and to decrease systems own infulence on a network, Cichlid uses special encoding based on state changes[9]. Cichlid is a real-time system. New measurements arrive all the time and all images generated by Cychlid are dynamic. In a Vertex/Edge diagram, Cichlid represents network in a natural way in a form of graph where vertices (network nodes here) are connected with each other with edges (links between machines). Each Edge provides extra information in a form of color, size and style.

2.1.4 SVision

A very special graphical approach is shown in a system named SVision. It is an intrusion detection system that uses 3-dimentional image to help determine hosts with abnormal behaviour. Not all services are controlled. Normally, the number of services, that are really critical, is limited. Different services have different activity and different expected load. Services, system takes care of, are depicted as service points on a service plane which is a circle. Hosts attracted most to the services they are using most. Because of some services are normally generate more activity than others, SVision uses anomaly factor which is a service dependent value. Initial position for every host is a center of the Service Usage Plane circle. When hosts are using services they will be attracted to them. To show differences between host load SVision uses 3-rd dimention. Each host is depicted with two different positions with respect to incoming and outgoing traffic. 3-dimentional space of SVision is divided into two subspaces, inbound activity subspace and outbound activity subspace. Spaces divided by the Service Usage Plane. Position of the host images in space is dependent of services it uses and the kind of activity. Hosts with dominance of inbound activity can be a machine under attack. This kind of hosts have inbound pictogram flowing up. Hosts with strong outbound activity can be an attacker which make outbound activity pictogram sink down on the graph. Hosts with balanced inbound and outbound traffic will generally stay close to the the middle of the graph, not so long from Service Usage Plane. Additional information can be achieved using spesial methods such as lines

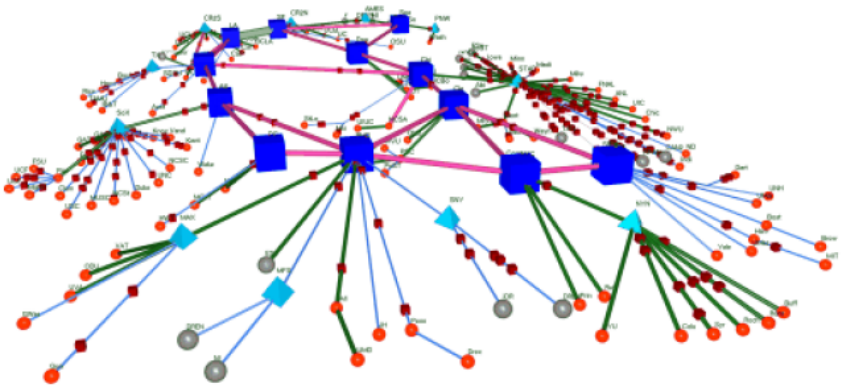

Figure 2.9: An example of Vertex/Edge Graph created by Cichlid[9].

from host pictograms and pictograms colors. Lines going from host in a form of short rays have different meaning for internal and external hosts. For internal hosts, lines related to the amount of open ports. Large number of open ports can be a sign of attack. In case of external hosts, lines related to the amount of IP addresses this host is trying to connect to. Different colors are used to separate internal and external hosts from each other. Intencity of color changes with respect to observation time. The longer host is observed, the lighter color is used. SVision is a distributed system. It consist of two components. The one that run on each controlled host and collects information. And central visualization program that receives information from sniffers and produce a 3-dimentional image of the network state. Filtering of significant information is done in the collection phase. The reason is that system should not produce high network load by itself. SVision can help to analyse data rolled back in time. Information can be either collected dyrectlty from network or it can be read from TCPDump[12] files. Using files recorded by TCPDump, administrator can analyse what happened with network in the past and find traces of suspicious activities[11].

2.1.5 Immersive Network Monitor

This system can visualize huge amounts of network data in real-time mode or do it in time perspective using roll-back. Immersive Network Monitor does not use complicated filtering system. It rely on human brain capability to recognize images. The system should just help user presenting information in way that is best suited for recognition by human brain. Immersive Network Monitor utilizes two modules for packet capturing. Both provides packet stream description based on source and target IP address, source and destination port number and type of protocol. Each stream can be registred with respect to start and end time. Or stream can be defined as a sequence of packets where time between packets is no longer than predefined value. To achieve better performance information collection and visualization are physically separated and run on different machines. Visualization show own network as a circle area on the plane. This territory is surrounded by a semi-transparent dome, which represents network borders or firewall. Hosts inside of home network are depicted inside of firewall shield. Other hosts located outside of shield. System transforms IP addresses of hosts into polar coordinates using special formulas. Internal Hosts located on concentric rings with respect to subnets they are belongs to. Streams of packets represented as lines connecting source and destination hosts. Lines from outside host goes in the direction from host to the center of the sphere. When line intercepts the

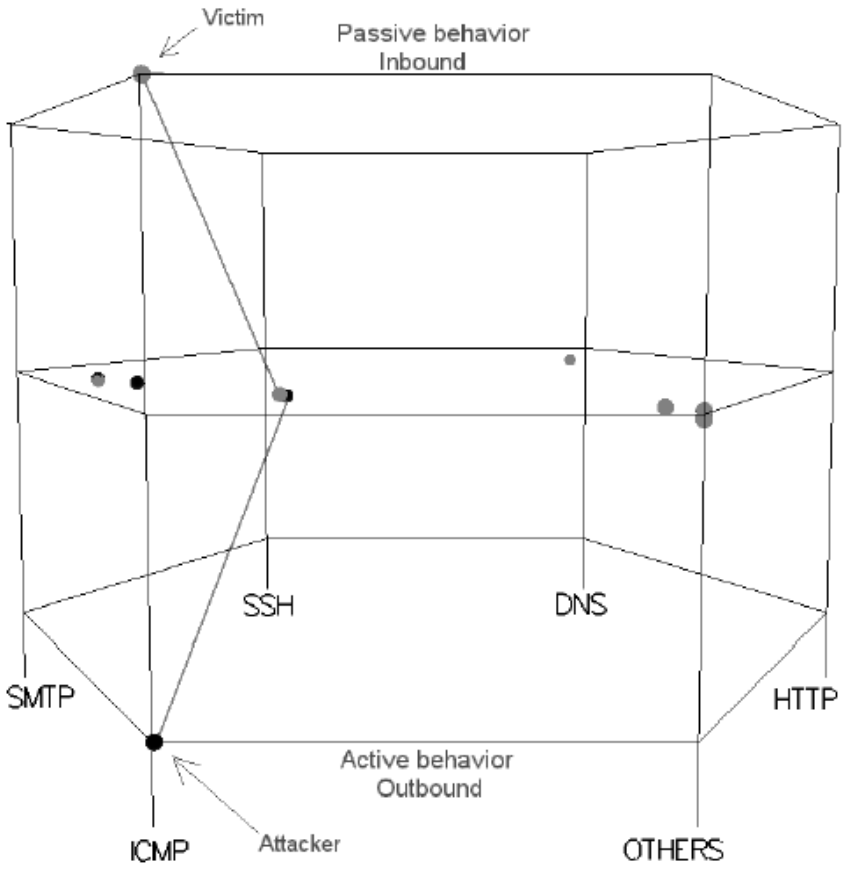

Figure 2.10: An example showing visualization of Ping of Death attack produced by SVision[11].

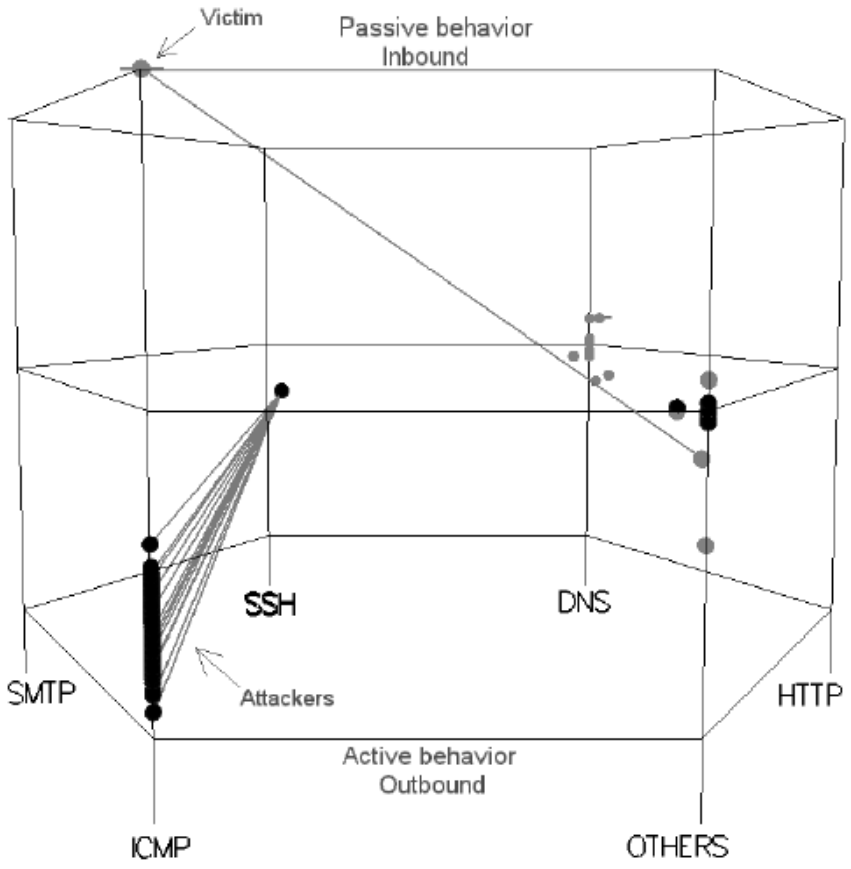

Figure 2.11: An example showing visualization of Smurf attack produced by SVision[11].

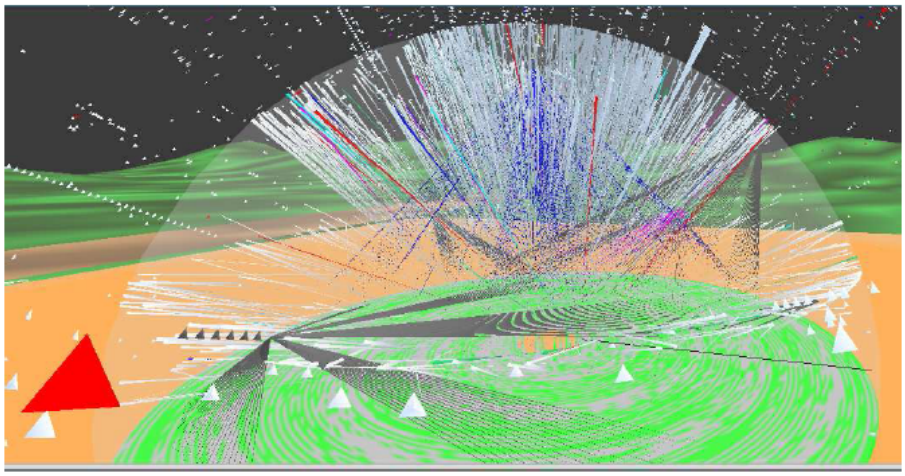

Figure 2.12: An example of image produced by Immersive Network Monitor[6].

firewall sphere, it changes direction and continue to the destination host inside. This makes possible to follow connections from one external host to one or more host inside of the home network. In addition to connection lines, system uses animated rays running from source to destination. Connection lines made of cylinders. Rays are made as cones with top pointing into packets stream direction alongs connection line. Providing of additional information is made using heads-up display. System uses colors to represent amount of sessions internal hosts involved into. Each dataflow ray can be colorured with respect to involved port-numbers. Operator can interact with object inzide of visuzlization, request more information about particular object of visualization, roll time back and forward using data collected in database.[6] As we can see, a system for effective analysis of data can be describes as following. Separate normal behaviour from unusual. Change from ”reading” to ”looking” which is more natural and effective for human beings. Add interaction to achieve more filtering and zooming on details. First, the whole picture, and then zoom into details. [13]MielogTudumi

(???) Let’s write a summary of all simple graphical methods such as lines and so on. 2.1.4

When graphs are huge, when workstation comes into trouble trying to visualize it, when operator can’t handle all information, then we need to think us a new solution. It is natural, that some complicated graphs can be logically divided into smaller parts. Then active parts of the graph can be represented in high detalization level. These parts operator is working with. Other parts can be represented with less details, and some distant parts of the graph can be replaced by basic objects like pictograms or icons. When operator changes focus to the distant object, the level of detalization will change and operator comes closer to the new object of interest. So we zoom in for more details and zoom out for overview the whole picture. Locating objects on different levels we can navigate this constraction as we go through the building with several floors[1].

2.1.6 ZUI - Zooming User interface

The main goal of any computer interface is to provide tools to control computing environment (both hardware and software). Most computer interfaces as we know them today are Graphical User Interfaces. All of them are based on principles that were first implemented

in Xerox Palo Alto Research Center and known as PUI[2] or WIMP[3]. These kind of interfaces are generally easy to learn and understand. On the other hand, there interfaces are tend to require lots of operations to do simple tasks. Operator needs to use time to manipulate interface, to organize it[2]. Some interesting concepts can be taken from a new kind of user interfaces, - Zooming User Interface. There is a one major problem in situations when we need to display relatively big amounts of information, the lack of screen space[14]. The possible concept here is virtual infinite workspace with possibility to zoom in and out, making different elements of interface visible for operator. Navigation in ZUI is done not only in traditional way of 2-dimentional Graphical User Interface. There is an extra dimention that can be used to zoom in to see particular element of interface or to zoom out to see the whole. In the beginning, operator can be presented a whole picture also known as Radar view. Next step is to zoom on particular element of interface also known as Zoomable view. In some cases it can be helpful to see both some specified object and the whole picture. So interface can be splitted and show different rates of magnification. Interface implements standard set of functions like zooming in and out, selection of object or group of objects, relocation objects into the viewpoint, scaling objects, defining relations between objects. Zooming operations change detalization level depending on how close object is to operator. ZUI give us a close connection between actions performed by user and data that interface is presenting[15]. An important part of ZUI design is definition of zooming or detalization levels based on properties of presented information. Users position along virtual Z-axis decides which portion of information will be visible and how much user will see. It also can be helpful using Semantic Zooming, which is a special kind of zoom showing different information or different detalization levels[15]. Following elements are the building blocks of a general Zooming User Interface:

- Render. An object that is representing terminal itself. It is responsible for scene generation and for interaction with user.
- At least one ViewPoint into 3D information space. Several ViewPoints can make navigation easier.
- Possibility to move inside of the scene.
- Constrain camera direction. The way user can see interface should be preprogrammed. Inside of Interface we have to be sure that it's possible to predict the way user moves and the way user can see objects.
- Position data into specific places within information space.
- Transform data according to users actions (changes in objects presentation in ZUI is natural, not so important for cdMagine, but, it can be a good idea to switch on and off some parts of graph or do something else)
- Define special control areas that are not conected to data but can be used for navigation purposes (remote controls, jump areas)
- Objects can be static or changable.

Characteristics of Zooming User Interface can be implemented using Virtual Reality Modelling Language with objects modelled and located in real 3-dimentional space.

[2]PUI stands for PARC User interface. It utilizes objects such as windows, menus, buttons and check boxes to create user interface. Mouse or touch-screen is used in addition to traditional keyboard

[3]WIMP is an alternative name for PARC User Interface concept. It is an acronym of Windows, Icons, Menus and Pointing Device.

Chapter 3

Presentation metaphors

Let us take a look on how we can use a 3-dimentional space to represent information. Let us assume that we have a set of diagrams of any type. Each of them shows a distribution of a particular parameter in the system. In the case of 2-dimentional interface we will be able to see one graph in full screen mode or a list or matrix of scaled pictures / pictograms or even list of icons. This organization of graphs / objects can be naturally implemented using ideas from Zoomed User Interface2.1.6. Same graphs can be located in a 3-dimentional space using following configurations[16][17].

3.1 Stack

Setting diagrams in stack. It's a simpliest way of combining graphs. Using a method we've seen from Tudumi2.1.1 we will place graphs that claims attention to the top of the stack, close to the operator. Graphs that system assumes not to be so important will be placed to the bottom of the stack. If backgrounds of graphs will be made of semi-transparent material, it will be possible to observe two or more graphs simultanously. Graphs can be compared in the same way as if they were placed on the same coordinate plane.

3.2 City

Graphical presentation based on City metaphor is similar to CityScape chart2.0.2. Placement of graphs is based on graphs' importance. Charts are placed into the front or to the back of the "city" depending of importance of each graph. Objects with similar importance can form streets along raws or columnes in an XY plane. Two different methods to evaluate importance can be used. When one will arrange diagrams in raws, another one can be used to arrange diagrams in columnes. In this case a graph located in the corner (1,1) will have the highest summary importance.

3.3 Solar System

Solar system is a kind of diagram that contains of charts located on the orbits around the center point. Initial ViewPoint is located in the center of the solar system. For most convenient interaction all charts can be wrapped into Billboard node. Billboard objects will constantly follow user, turning around to achieve best observation angle. Default rotation is a rotation about Y-axis, but it's possible to configure Billboard such as it will follow user in all kinds of movements. It's possible to design a system such as all graphs can be observed by only turning around. When the orbit of the graph is too long and graphs can't be observed in details from the center of the system, then it can be helpful to place Standard

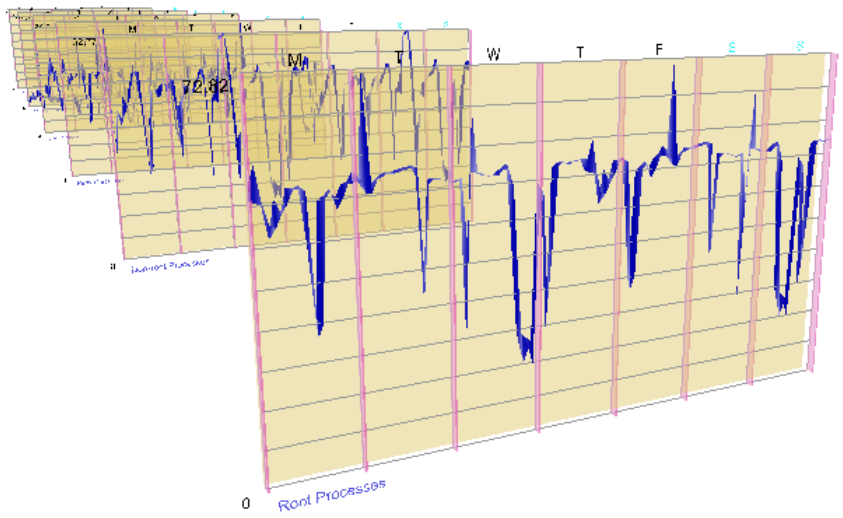

Figure 3.1: Stack Metaphor. Scene generated by cfMagine prototype

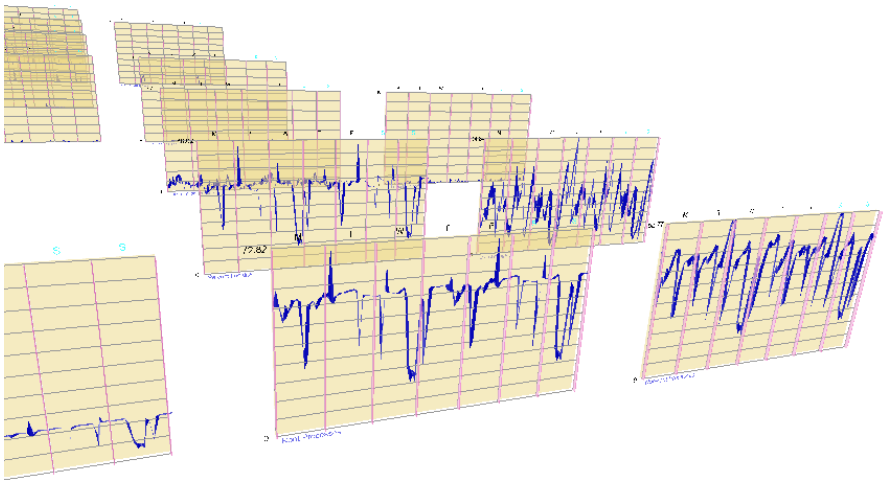

Figure 3.2: City Metaphor.

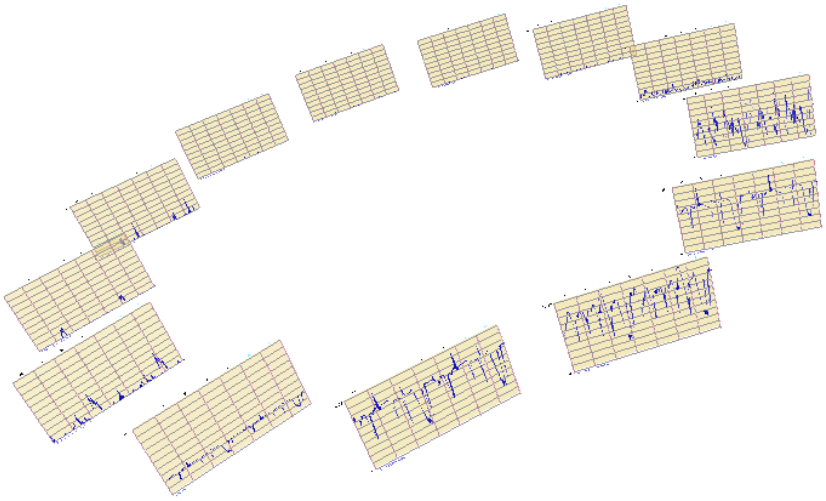

Figure 3.3: Solar System Metaphor.

ViewPoints on the orbit between graphs and center. Direction of each ViewPoint should be calculated in a radial direction from centers to the charts. Importance of each graph can be used to decide if the orbit of object will be closer to the center or not.

3.4 Spiral Galaxy

In case of Solar System, we group charts accordingly to their importance. Similar importances will be placed on the same orbit. If we arrange diagrams according to importance and then will place them into the space changing angle and radius of the orbit, then we will get a structure of Spiral Galaxy.

3.5 Tower

In a Tower metaphor, the placement of charts is similar to the Solar System. But less important elements are not placed to the outer orbits. The elements are placed on the floors of the Tower with most important on the ground floor, where user is located in the start, and less important graphs are located on the higher floors. Operator can navigate turning around and sliding up and down if *VRML*-browser supports this feature.

3.6 Sphere

The last metaphor we will take a look on is a Sphere. Objects are located on the sphere with front sides turned to the center where operator located in the beginning. More important graphs are located close to the equator of the sphere. In addition, elements can be placed close to zero-meridian, so we can combine two different importances.

All represented metaphors can give an overview over all graphs with the right placement of the Initial ViewPoint. That's the first part of the Visual Information-Seeking Mantra2, we taking an overview of the whole system.

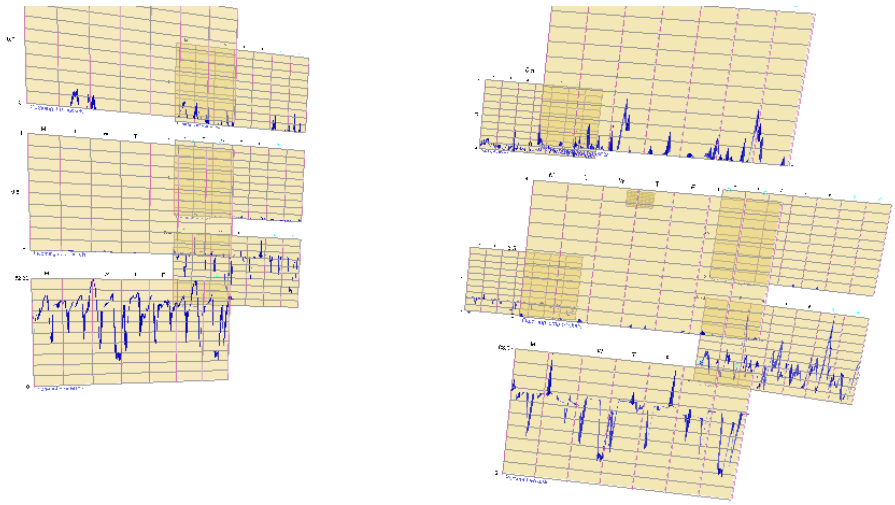

Figure 3.4: Chart diagrams placed according to Tower Metaphor. Users' ViewPoint is located outside of the Tower to provide a better better picture of the Tower structure.

Chapter 4

Virtual Reality Modelling Language

Let us see what is the basic requirements to a system that can be used to desigh a simple interactive 3-dimentional environment?

- It have to provide simple geometrical shapes such as lines, polygons, blocks, spheres, cylinders and so on. Possibility to handle text is a greate advantage. It must be able to apply color or/and texture to each simple object.
- There must be methods to manipulate objects, such as change position and orientation, change color and visibility of objects and scale objects.
- We need a possibility to group objects and reuse objects descriptions.
- System should provide methods to explore objects, methods to support animation, triggers and sensors to control virtual scene and methods to change the World accordig to external events from both human operator or other system.

Designing 3D application needs special skills. 3D programming is not trivial and time-consuming. The World have to know all about objects, their properties and configurations, know how to draw them, know how objects can interact with each other and with user. Most applications are platform dependant and the grade of dependency is even higher for applications that work with 3-dimentional graphics[7]. This situation was changed with invention of Virtual Reality Modelling Language or *VRML*. It was originally designed to deliver 3-dimentional Virtual Worlds over the Internet. As many technologies connected ti Internet, *VRML* have following great advantages:

- It's an international standard. Descriptions of standard can be found in ISO:14772-1:1997[18] and ISO:19776:200x[19]. Web3D consortium[1] is the organization behind development and support of the language.
- The language is really simple. Files are text documents. *VRML97* uses Brace-and-bracket utf8 style. Next generation of language, *X3D*, is a subset of XML.
- Language is portable and can be used on any kind of computer platform that has software supporting *VRML* format.
- *VRML* contains both data and description of presentation.
- *VRML* uses the same content delivery principle as HTML (server - client paradigm).

[1] http://www.web3d.org/

First version of *VRML* comes from Silicon Graphics' Open Inventor, which was also known as IRIS Inventor. Open inventor is an object-oriented toolkit written in C++. It provides a set of 3D objects and methods of manipulation and interaction of objects with each other and with user. The toolkit was designed with portability in mind and could be used on different platforms. Basic objects in modern *VRML* and *X3D* are still the same as they were in Open Inventor[20]. First specification of language, known as *VRML* 1.0 was released in 1994. Interaction possibilities in *VRML* 1.0 were limited and it was redesigned in 1996. *VRML* 2.0 specification was released by *VRML* Architecure Group in 1996. Static worlds were enhanced, interaction and animation with scrypting made dynamical worlds possible. New objects could be prototyped and reused many times. To produce more real scenes, *VRML* 2.0 have got possibility to integrate sound, add fog to the sceene, and create complicated surfaces with objects of type elevation grid. Sensors were added to increase possibilities for interaction. Interpolation node was added to introduce animation.

VRML uses description to create 3-dimentional models of objects in virtual space. Special software needed to visualize Virtual World. *VRML* can be seen as an extension for World Wide Web and both server and client are the same as in standard Web-browsing. *VRML* scene is scalable and can be seen from different points of view and under different angles. *VRML* scene can communicate with other components using External Authoring Interface.[21]

Let us take a look on how *VRML* works. *VRML* uses well-known client-server type of communication. Client software requests World description from the server. Client side (web browser) need a special component (VRML browser) that can understand *VRML* format and produce an image of virtual scene. Some *VRML* browsers can operate in both plug-in and standalone application mode. World description can be red from local disk as well. *VRML* accepts use of MIME[2] to include different kinds of information into one file. As a universal standard, *VRML* accessible from virtually any platform. It doesn't matter what operating system or browser are in use, since *VRML* is ment to be cross platform and universal in the same way as HTML. Simple and clear syntax makes it easy to generate *VRML* files[22]. Files can be easily edited using any kind of text manipulation program. Files can be easily compressed with relatively good compresstion ratio. When standard came into the World for the first time, it was probably misunderstod and misused. *VRML* is used to create simple objects from the real world. AAnother interesting example is architectural models. These models can show buildings in details from both outside and from inside. Users can walk, look around and even manipulate with furniture. In the same way *VRML* is used to model terrain with infrastructure and buildings. *VRML* has strong positions in chemistry especially in molecular modelling. Modelling of mashines is another area where *VRML* technology is widely applied. *VRML* can be used to create 3D games as well. But more important for us is that *VRML* can and must be used to create users interface and provide a new way of access and manipulate data. If we compare *VRML* scene with for example HTML page containing raster graphics images, then we can see that in many cases that *VRML* can be smaller in size becase of vector nature of graphics and possibility to reuse object descriptions again and again. When *VRML* scene is downloaded into *VRML* browser it's still remains dynamic and functional, user can change point of view and manipulate with objects.[5]

To view a *VRML* world we need a special program called *VRML*-browser. A main function of *VRML* browser is to produce a Virtual World according to description from .wrl or *.X3D* file. Browser provide possibility to walk inside of the World, interact with the World and control what happen inside supporting sensors, routes and so on[22].

The next generation of *VRML* is Extensible 3D Graphics or *X3D*. It has all features of *VRML*97 and clearence of XML[3]. *VRML* was designed in a time when modems were dominating and size of files was really important. It is not the case today. Files of new format

[2]Multipurpose Internet Mail Extensions

[3]Extensible Markup Language

```
NavigationInfo {
  type [ "EXAMINE" "ANY"  ]
   # transitionType [ "ANIMATE"  ]
}
Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0.1 0.8 0.5
    }
  }
  geometry Box {
     size 1 3 5
  }
}
```

Figure 4.1: Example of *VRML*97 code.

can be slightly bigger than *VRML*97 files, but *X3D* has the power and elegance of XML. *X3D* Systems provide backward compatibility with *VRML*97 standard. Any *VRML*97 file can be translated to the *X3D* format without any loss in objects or functionality. Additional features of *X3D* are XML integration, multi-texturing, NURBS[4] and new scripting API. In addition to 3D objects *X3D* has a special set of 2-dimentional components (Arc2D, ArcClose2D, Circle2D, Disk2D, Polyline2D, polypoint2D, Triangleset2D)[23]ISO:19776:200x Because of it's conection (inheritance) to XML, *X3D* has a great possibility to structure data, it's easy to read for both human beings and machines, validity check can be done simplier and faster. It is possible to separate data and a way of presentation, such as presentation can be changed when data is still the same. It's platform independent and has good chances to be supported widely[24]ISO:19776:200x. *X3D* is still under development. To describe 3-dimentional scene, one can choose any of following formats:

- *VRML*97 syntax. These text files are written using Brace-and-bracket utf8 style and have extension .wrl.
- *X3D* format based on XML. These files have *.X3D* extension.
- *X3D* based on Brace-and-bracket utf8 style. These files uses synaxes known from *VRML*97 and objects from *X3D*. Extension for this type of files is *.X3D*v
- *X3D* "Binary" encoding. The aim of creating this format is to apply compression on 3-dimentional data to shorten the transmission time of models and Scenes across a network and to add data encryption to protect sensitive model information.

X3D code based on XML is generally longer. Most objects in XML style created by pair of tags, that creates overhead. Code can be easily converted from *VRML* to *X3D*. Backward conversion is possible if objects defined in both standards are used.

X3D standard described in ISO/IEC FCD 19776:200x[19].

Earlier both Microsoft™and Netscape™provided *VRML* browsers bundled with their Web-browsers. But, it's not the case anymore and users have to choose and install *VRML*-browser self. Some browsers support both *VRML* 97 and *X3D* formats. Some older programs can work with *VRML*97 only. Here is a short list over browsers that can be installed and used. There is another technology that can be used to present information in effective way. It is a SVG or Scalable Vector Graphics. As VRML and X3D, *SVG* is

[4]NURBS stay for non-uniform, rational B-spline. It is a mathematical model used in computer graphics for rendering curves and surfaces.

```
<Scene>
  <NavigationInfo type='"EXAMINE" "ANY"' transitionType='"ANIMATE"'/>
  <Shape>
    <Appearance>
      <Material diffuseColor='0.1 0.8 0.5'/>
    </Appearance>
    <Box size='1 3 5'/>
  </Shape>
</Scene>
```

Figure 4.2: Example of *X3D* code based on XML format.

also an open standard. It is created by W3C[5]. *X3D* already has subset of objects from *SVG*[X3DIntWeb**?**]

4.1 Browsers

Xj3D[6]. Is a project driven by web3D consortium[7]. It supports both VRML97 and X3D formats. Xj3D is written in Java. Source code is acceptable. Xj3D is licensed under GNU LGPL[8]. Programmed in Java it must be possible to run it on virtually any computer platform. Installers for Windows, Linux, Solaris and Mac OS X are available.

Flux is a high performance VRML and X3D browser. Designed by MediaMachines[9] Flux Player is available for free for personal use. Can be run under Microsoft™Windows™only.

Octaga Player is a powerful VRML and X3D client designed by Norwegian Octaga AS[10]. Player is freely available for personal non-commercial use (with banner screen and logo that can be turned off with obtaining license). For commercial use license must be obtained. Version for Microsoft™Windows™is the one that is under development. An older Linux version with reduced functionality is still available.

BS Contact VRML/X3D. Designed by Bitmanagement Software GmbH[11]. Supports both VRML97 and X3D. Commertial software that can be downloaded for testing / evaluation purposes. BS Contact works under Microsoft™Windows™.

Cortona is a VRML plugin designed by Parallel Graphics[12]. It supports VRML97 standard only. Functionality of this browser is extended with additional nodes and capabilities that are not presented in VRML97 standard. VRML files designed using Cortona extensions can not be rendered using other browsers. Cortona is free for testing and non-commertial use.

FreeWRL[13] is an Open-Source browser supporting both VRML97 and X3D. It can be run under Linux and Mac OS X platforms. Is licensed under GNU LGPL[14].

Cosmo Player was one of the most popular in a VRML time. It was initially designed in Cosmo Software, that became a part of Silicon Graphics, Inc.[15] and was sold to Computer Associates[16] after all. Cosmo Player is no longer supported. It works with VRML97 files

[5]World wide Web Consortium. `http://www.w3.org/`
[6]http://www.xj3d.org/
[7]http://www.web3d.org/
[8]GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
[9]http://www.mediamachines.com/download.html
[10]http://www.octaga.com/
[11]http://www.bitmanagement.de/
[12]http://www.parallelgraphics.com/
[13]http://freewrl.sourceforge.net/
[14]GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
[15]http://www.sgi.com/
[16]http://www.ca.com/

only. Versions for MS Windows, SGI Irix and Machintosh classic are still available from Intrenet[17]. Versions for IRIX 6.5 and later are still available from SGI. Cosmo Player was reincarnated once as Pivoron Player from Nexternet with additional support of X3D, but it did not succseed.

Orbisnap. Orbisnap[18] is a free VRML97 compatible browser designed by HUMUSOFT[19] s.r.o. It can be run under Windows, Linux 32/64-bit, Sun Solaris, Mac OS and HP-UX. Browser implements standard set of actions (Walk, Fly, Examine). It has an interesting feature, - "Define ViewPoint". Users of Orbisanp are allowed to define their own viewpoints when they browse VRML world. The world with can be saved including new user defined ViewPoints.

4.2 VRML File format

X3D file begins with header that is similar to header used in XML. It contains descriptions of language, document type, profile and additional meta-information. Virtual World description located between tags ¡Scene¿ ¡/Scene¿. Objects inside of Virtual World organized in a tree structure. The Scene object is on the top of the hierarchical tree. All other objects are children of main scene. Objects can be organized in groups and groups can contain groups and so on. Almost all objects have fields containing properties. For example Virtual Scene has parameters suporting Background and Global Sensors[22]ISO:14772-1:1997. A simple brick and a genuine part of the Scene is a node. To avoid huge and complicated files, standard has an option to include external descriptions into the world using Inline node, which simply include description from external file into the main World description. Position and appearance of each object inside of the world can be set using standard transformations like translation, rotation and scaling. Using textures and simple shapes can decrease file size and will make scene look more real and appealing[25].

Folowing objects classes presented in VRML and X3D and can be used for design of 3-dimentional scene:

- Simple shapes. It is a set of simple geometrical objects. Based on supernode Shape these object can be presented in different forms such as Box, Cone, Cylinder, ElevationGrid, IndexedFaceSet, Indexed-LineSet, Sphere, Normal PointSet, Text.
- Appearance nodes are responsible for visual presentation of objects. These nodes can define Color, FontStyle, ImageTexture, Material and some tranfromations applied to Textures.
- Grouping nodes allow us to group objects together in VRML. It make manipulation with objects much easier. One of the most often used nodes Transform (including translation, rotation and scale) is functioning as grouping node as well.
- Environment nodes are responsible for appearance of 3-dimentinal scene. These includes Background information, light sources, Fog and even surrounding sound
- Transformation / Animation nodes is a set of Interpolators that can be used to manipulate properties of objects, position, orientation and control time.
- Interaction / Sensors is a set of nodes responsible for interaction with user.

Full list of nodes with detailed information can be found in VRML and X3D ISO standards ISO:14772-1:1997[18] and ISO:19776:200x[19].

Simple geometry nodes such as Box, Cone, Shpere and Cylinder have only outside surfaces. When viewed from the inside the results are undefined.

[17] http://www.karmanaut.com/cosmo/player/
[18] http://www.orbisnap.com/
[19] http://www.humusoft.com/

Let us take a closer look on some nodes that can be useful for our project.

Coordinate system inside of the World is Euclidean coordinates system. X-axis and Y-axis are the same as in 2-dimentional World. X-axis goes from left to right and Y-axis pointing up. Z-axis goes in a direction from the screen to operator. Point $(0, 0, 0)$ is located in the middle of the screen. All angles measured in radians. To change a position, orientation or size of any object in the World we have to do a Transformation that exist in 3 different types. Transform translation takes 3 coordinates to change the position on child node/nodes. Transform rotation takes 3 coordinates that define rotation vector and angle value in radian. Transform scale takes 3 values that defines scaling along each dimention (x, y, z)[23]. Shape node is the node that contains subnodes that describe geometry and appearance of objects. Material subnode describes appearance of the object (such as color). Subnodes for box, cylinder, sphere, PointSet, IndexedLineSet, IndexedFaceSet, Extrusion and Text are responsible for geometry of object. One of the important parts of the world is the stack of ViewPoints. ViewPoints can be imagined as cameraes located in particular places inside of the Scene. Each ViewPoint has position and orientation defined. All ViewPoint are collected in stack such as user can switch between them. Most VRML browsers provide possibility to shift through the stack in step by step mode or directly choose the ViewPoint user needs. To see in the darkness we need light. By default there is only one source of light in the VRML Scene. It is the "head-light" of the user, the directional torch located on the head of virtual user. All other light sources must be described in VRML-file. Colors in VRML is traditionally defined in RGB format, giving color as a mix of intencity for Red, Green and Blue components. This way of coding give us white color when color vector is set to (1 1 1) and black for color vector (0 0 0).

There are 3 standard types of lighting source we can choose from:

- DirectionalLight. This source of light can be compared to our Sun. System assumes that rays of light come from the source located on an infinite distance from our Scene. Because of this, all rays of light are perfectly parallel to each other. We can decide direction of light (using direction vector), color, intencity and some other properties. A special feature of Virtual DirectionalLight is that this source will light only the objects of the same parent grouping node. So, objects located outside, in another grouping node, will not be highlighted. Intencity of DirectionalLight does not change with distance. Following code is used to describe DirectionalLight

  ```
  <DirectionalLight direction='1 0 -0.707' on='true' />
  ```

- $PointLight$. This source of light is located inside of the Scene. PointLight lights in all directions. It is such a kind light bulb. In addition to standard properties like color and location, PointLight has a radius. This property defines the radius of the sphere where the light from PointLight can reach. The intensity of light from PointLight can change with the distance if $attenuation$ property is defined. $PointLight$ has global effect and will light all objects around independently from grouping node. Following code example creates a PointLight source identical to the Sun

  ```
  <PointLight color="1.0 1.0 0.9" location="0.0 1000.0 0.0" radius="10000.0"/>
  ```

- $SpotLight$. This light source has location and direction. In addition, there are two angles that help to define properties of the SpotLight. BeamWidth angle describes an agle where intencity of light is constant. Cutoffangle is an external angle greater than beamWidth angle. Area outside of cutoffAngle will not be enlighten, intensity of llight there equal zero. Area between cutoffAngle and beamWidth will be filled with light with linear degradation of light intencity from beamWidth with top intensity to cutoffAngle with intencity of 0.

```
<SpotLight location='6 0 5'
  direction='0 0 -1'
  radius='6'
  on='true'
  color='1 1 1'
  intensity='0.5'
  ambientIntensity='1'
  attenuation='1 0 0' />
```

Rendering og $DirectionalLight$ goes faster in general, because this calculation includes the only nodes in one particular group when $PointLight$ and $SpotLight$ are global in nature[18]ISO:19776:200xVRMLPostProd. All transformations done with parent nodes will affect all types of light. In case of complex objects, one should remember about shades. Geometry nodes kan be made of material with following color characteristics affecting object presentation

- $ambientIntensity$, which defines how many ambient light from light sources object should reflect. Ambient light doesn't depend on direction of light and will be reflected in all possible directions.
- $diffuseColor$, which is the color that combines with texture on the object. This field is responsible for reflection of light with respect to it's direction.
- $emissiveColor$, which defines the color object emmits itself. It makes object glowing in the dark. So, we have a possibility to make objects self-lightning, without need for any external lighning sources (we can even turn off the default headlight). Using this option can be really helpful when one works with model before light sources are set on its places. In combination with TimeSensor, we can create an object that blinks[18]ISO:19776:200xVRMLPostProd.
- The $transparency$ field defines transparency of an object. Objects with transparency equal to 1.0 became absolutely transparent. Objects with transparency equal to 0.0 will be completely opaque.

Ambient colour of the object is calculated as $ambientIntensitydiffuseColor$.

Two powerful functions of VRML, DEF and USE can be used with any objects or group of objects and with objects properties. The first one DEF is used when object define for the first time. Then object definition can be used many times. As a result VRML-file will be smaller in size and easier to understand and maintain[26]. (???) Simple example of saving size on using DEF/USE for color in let say 50 objects. What we will spare on this operation?

Chapter 5

Navigation and interaction

5.1 Navigation

Navigation and interaction are essential parts of VRML. Standard input devices such as keyboard and mouse are used to navigate and interact with Virtual World. The navigation is more complicated than 2-dimentional navigation. 3-dimentional World can be seen from different viewpoints and under different angles.

Following types of navigation can be used in VRML

Walk, as it's easy to understand from the name of this mode is just a natural walking inside of VRML World. Gravitational field will hold user on the plane of Z- and X- axis. User can go forward and back, turn to the left and to the right, look up and down. Fly navigation mode simulates the flight in the scene. Some browsers provide additional navigation modes like tilt (turning user up and down and to the left and to the right) and Pan (sliding in vertical and horizontal directions). Examine mode allow to turn the whole World as an object and zoom in and out scene.[22]X3DFund

An important part of navigation is to get all possible information about objects. When we move unsignificant data far away from operator, we do a filtering of information, setting focus on datasets that are important. That's the second part of the Visual Information-Seeking Mantra2, we filter to get the most important result. VRML provides rich tools for interaction with user. Additional information about object can be given when user clicks on particular object, when mouse is just located over a particular object or even when user is close to object (based on proximity sensors)[27]Worlds.

When VRML-browser interacts with user all movements can be done in a form of smooth movement or discrete jumping. Direct jumping can make users disoriented, especially when user is new to the system. Direct jumping can save some time when the person is familiar with system and environment[23].

A powerful and effective method of navigation in a VRML World is based on use of ViewPoints. Carefuly defined sequence of ViewPoints can make navigation much easier and save operator from the danger of being lost in 3-dimentional space. Syntaxis used to define ViewPoints is simple

```
<Viewpoint description="Start" orientation="0 0 -1 0" position="0 0 120"/>
```

Position describe a physical position of ViewPoint as X,Y,Z coordinates. Orientation defines vector in which direction "camera" will be turned to. The last value for orientation defines rotation angle for "camera" and should be in radian. When we define ViewPoints for different presentation methaphors3, we should follow some simple rules. Let us define distance of optimal view a as a distance from defined ViewPoint to the object, ViewPoints' camera points to, where the objects' image fills angle of view of the ViewPoint. When operator changes ViewPoint, it will be show particular object in a best possible way. For

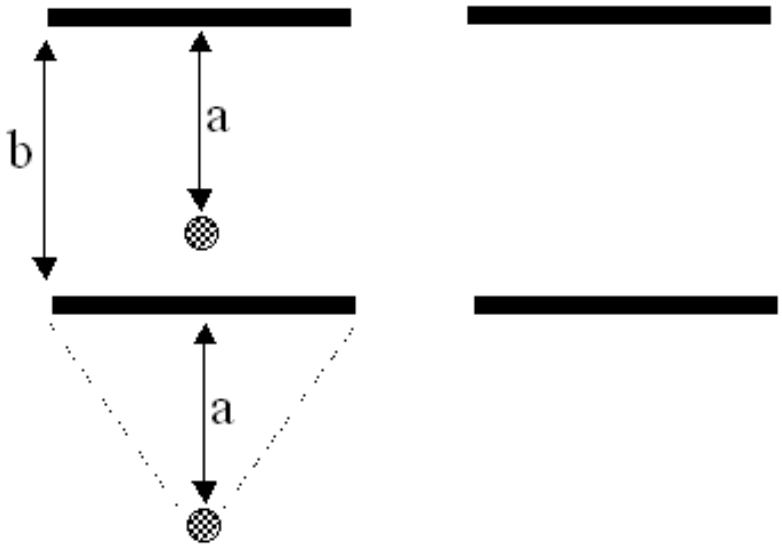

Figure 5.1: *ViewPoints* location in case of *City* metaphor. Distance between objects b should be greater than the distance of optimal view a

metaphors such as Stack3.1 or City3.2 distance between objects of presentation b must be more than distance between ViewPoint and object a. Metaphors like Solar System can use several schemes for ViewPoint location. Most basic is to place an initial ViewPoint to the center of the Solar System. User can easily observe objects turning around. In a situation where the radius of first orbit R_1 is more than distance of optimal view, ViewPoints can be located on the lower orbit with radius $R_v = R_1 - a$. In the same way, ViewPoints can be located on the outer orbit with radius $R_v = R_1 + a$. In this case all objects should be turned outside.
Another approach is to slide the center of the ViewPoints' orbit against the objects orbit. Sliding distance is equal to the distance of optimal view a. All object should be turned in the same direction or wrapped into Billboard node. All ViewPoints are located on the same orbital positions as objects and have the same direction. Named strategies of ViewPoints placement can be used for other metaphors such as Spiral Galaxy3.4 and Tower3.5 as well.

5.2 Interaction

Additional information inside of the World can be povided dependent on users actions.

5.3 Anchors

A simpliest way of interaction is binding links to objects. Every object or group of objects can act as a link in a same way as hypertext links in HTML.

```
<Anchor description="Test Anchor"
  url="http://mindcrawler.net/VRML/Anchor_Test_01.{\em X3D}#Point2">
  <Shape><Box/></Shape>
</Anchor>
```

Any kind of object can be a target. It is possible to define a starting ViewPoint for target file, such as user will be transfered directly to that particular point. Anchore node works

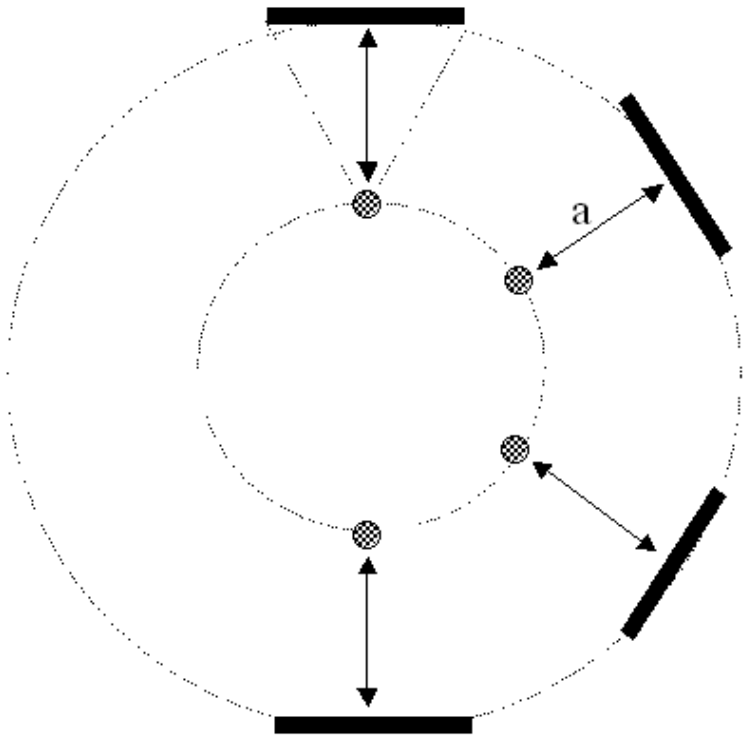

Figure 5.2: *ViewPoints* location in case of *Solar System* metaphor. *ViewPoints* are located on the lower orbit

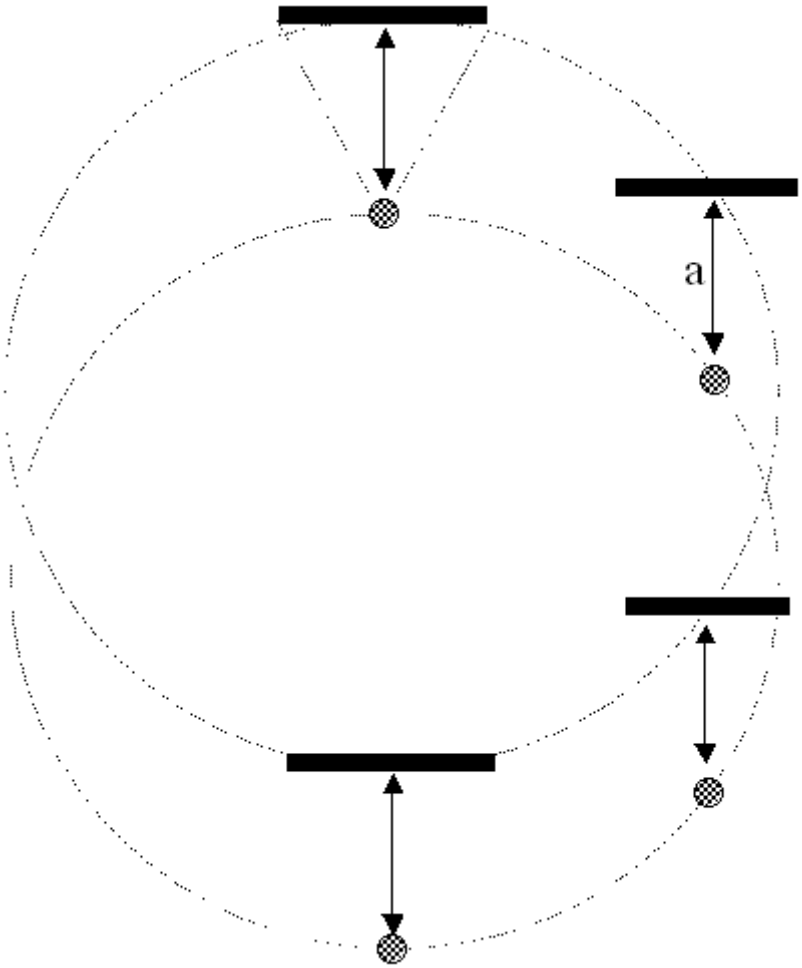

Figure 5.3: *ViewPoints* location in case of *Solar System* metaphor. Centers of orbits for objects and *ViewPoints* are slided against each other

like grouping node. All objects described between tags ¡Anchor¿ and ¡/Anchor¿ can be clicked on and will activate defined link.

5.3.1 Sensors

Another methods of interaction include different kinds of Sensor nodes. Some Sensors must be activated with mouse click. These called Pointing Device Sensors. Some of them can just feel that mouse pointer is located over them. These nodes has different shapes to fit all kinds of needs.

- CylinderSensor. This Sensor simulates rotation of an invisible cylinder. The axis of rotation is the same as Y-axis of local coordinate system. Sensor is binded to the shape of its parent node. Cylinder sensor active when mouse button is pressed down.
- PlaneSensor. This sensor maps movement of the pointing device into the plane parallel to the plane build on X-axis and Y-axis. Sensor can be activated in zone that is similar to geometry of sensors parent node.
- SphereSensor. This node is similar to CylinderSensor. It provides rotation about the origin of current coordinate system.
- TouchSensor is a special kind of Sensor. It doesn't thet user pushes the mouse button. TouchSensor can feel that mouse poiter is over the parent node of TouchSensor. Active area of Sensor is similar to the geometry of parent node. In addition to movements, TouchSensor can detect when user pushes and release mouse button. This can be used to move objects and produce "drag-and-drop" effect.

Some other Sensors can control position of user inside the World. Position change can be a trigger that activates objects or actions. This task is controlled by following sensors:

- ProximitySensors. These Sensors can generate events when user crosses borders or moves inside of defined Proximity Area. Both movements and rotations are detected. Sensor breaks events generation when user leaves Proximity Area. Sensor has a Size field that defines bounds of Proximity Area box. If Proximity Area defined big enough (let say, big as World), then it can be used for generate events on every movement of the user. The Area surrounding the whole World will detect user entrace immediatly after World-file is loaded to the Browser. This can be used to do initial automation.
- VisibilitySensor. This Sensor based on sensitive box of defined size (the same kind of box as in ProximitySensor). VisibilitySensor can detect if user can see this box or not. If Sensor box defined around some objects, then Sensor can detect if these objects are visible for user or not. Using these kind of sensors can improve performance of scene rendering. Objects outside of users field of view can be striped for details, animation can be turned off and light sources can be switched off[18]ISO:19776:200xVRMLPostProd.

All Sensors can be Enabled or Disabled. To make rendering faster, we do not need to control complicated shapes. We can simply use ProximitySensor with simple shape. It makes rendering faster, beause browser need to do less computations in this case[22].

Anchor node is a grouping node and it will affect all child nodes between $< Anchor >$ and $< /Anchor >$. Sensor nodes are children nodes under the same grouping node as geometry objects that sensors are connected to[18]ISO:19776:200x.
To increase the amount of information acceptable to user and to provide more comfort, we can use a special kind of objects, - HUDs (Heads-Up-Display's). These objects can travel continuously with user. Heads-Up-Display can contain information and extra controls, for example links activating "jump"-function to a special ViewPoint. Proximity sensor set at

the origin and generates events all the time user change position or orientation. HUDs are really good to provide additional information on the fly[5]fisk-immersive.

When the Scene is huge and browser gets hard load rendering lots of graphs, we can use of LOD level of detalization to change the face of graphs far far away. The term means that there is no need to render objects that are far away from users current position. Such a distant objects can be replaced by simplier models. Object state will change to the original form when user is close enough or when transformation is initiated by trigger/sensor[5].

Chapter 6

cfEngine

6.1 script vs GUI

Configuration and maintainance tasks can be done in a different ways. Some issues can be solved using shell scripts. Some can be done using programs designed for Graphical User Interface. Both methods have their own disadvantages. Shell scripts are tend to be platform dependent. It is common case that script have to be modified before it can be run in a new environment. Writing universal scripts that can be run on different platforms is not a trivial task. Programmer should be familiar with several systems, programming involves plenty of testing, script code should be designed using lots of control sequences checking environment, locations and variables. Syntax of shell commands and objects locations can differ from system to system. Tools based on Graphical User Interface are good for novices. The problem here is that these tools demand operator to participation in maintaining process. Working with GUI-based tools is a really time consumig task. GUI-based tools are poorly suited for tasks that should be repeated on the regular basis or tasks that should be done automatically without attention from system administrator. It's well known fact that experienced system administrators prefer script-based solutions.

6.2 Configuration Engine

cfEngine was born from shell scripts. With the power obtained from scripts and with elegant adaptable configuration language, cfEngine is a good solution for administration purposes[28]. cfEngine or Configuration Engine is a system administrator tool that can be used to perform tasks of installation and maintainance of computer systems. Among these tasks are control and maintaining of filesystem permissions, setuid root programs, symbolic links, automatic configuration of network connections, preprogrammed editing text files, doing garbage collection, executions of scripts and some other tasks. Let's take a look on components of cfEngine

- cfagent is a main component of the cfEngine. This program is the one that is doing maintinance and configuration job according to instruction in configuration file.
- cfservd is a component with two functions. It function as a file server and can start cfagent remotely. Because both operations are critical, cfservd uses RSA authentication and control of IP-address
- cfexecd is a cron-like deamon. In addition to scheduling function, it can send e-mail (for example it can send output from cfagent to the administrators' mail-box).
- cfrun is designed to contact cfservd on remote machine and to start cfEngine on it

- cfkey is a tool that generates public-private keys used for remote authentication.
- cfenvd is an environment detector component that collects information about host
- cfenvgraph is a tool used to extract data collected by cfenvd

cfEngine is a kind of ”Immune System” which means that cfEngine learns all the time and able to protect computer system in the same way as biological Immune System do. It can remove garbage from disks and memory. It learns about new threats and can distinguish whether the system is healthy or not. It repairs damaged parts of the system. All these operations should be done by host itself, without assistance from operator. Host is responsible for holding itself in healthy state and will ask operator for help only when ”immune system” can not fight the problem properly[29]feedbackEvalImmUnixNL. System can be used to configure both single hosts or lots of hosts in a heterogenous networks as well. cfEngine uses a special language to define configuration and maintainance tasks. Language is not a kind of programming language, but a description of how the particular system should be configured. Administrator writes a single file to describe configurations of different hosts or groups of hosts in a network. This configuration file is distributed to each network host. Each machine will run it's own configuration procedure, based on instructions from configuration file that matches to the that particular machine. cfEngine uses the idea of classes. Classes in cfEngine contains data and methods. Definition of class can be done dependent on hardware architecture, host related information (such as hostname) or time (for example day of the week or hour of the day). Classes can be grouped together and any group can be a class itself. Workflow in cfEngine can be fully controlled by administrator. During runtime cfEngine obtains information about the system it runs on. Then it parses configuration file extracting actions that are suited for the host it runs on. Then cfEngine checks once again if everything is correct and all required information is available. To be sure that configuration and maintaining commands can be done properly, system will do additional security, network interface and filesystems availability check. And then configuration and maintinance tasks are done in a predefined way. Actions can be done on regular timed basis (such as running task each friday night) or when it's needed (run garbage collection when free space is is less than 10%). Changes in central configuration file can affect all hosts in network or just a single particular host as well. When cfEngine is run on a machine with correct configuration, it will only control the configuration. cfEngine will not apply any changes if it is not needed. cfEngine is not a closed system. It can invoke other programs to do job or it can be invoked by other programs when they need assistance cfEngine can provide. [30]paper1paper2CompImmunfeedback. Computer systems are influenced by users or other computer systems. Some of these interactions repeat on regular basis, some are more random prosecces. Finding activity patterns can be useful in learning about computer system. It can help to predict systems' behaviour in the future. It can provide assistance in detecting suspicious events[29]UnixNL. If we want to understand computer system more deeply, we need to do an analysis based on long time period. System state can be described using a set of variables (such as CPU load, free place on Hard Disk and so on). These data can be used in a statistical analysis of the comuter system[31].

6.3 Collection of information

Using entropy we can analyse systems' state and predict systems' future. For example, low enthropy for disk usage variable means little usage. High enthropy shows high load on disk subsystem with lots of data transfers from and to disk. Measurements of entropy for source IP-addresses for incoming packets can show what kind of processes happen in the network. High entropy means that we are receiving packets from many different IP-sources. Low entropy is a sign that there is only few machines communicate with us. It seems to be most natural to do analysis on a weekley basis, scince an activity of human-computer

system has strong connection to the time of the week.[32]feedback. One of the components of cfEngine, cfenvd is a deamon that is collecting information about system state. This deamon is absolutely silent and is doing its' job in background. Information is stored in the Berkeley Sleepecat database[33]sleepycat. cfenvd uses network packets capturing utility, tcpdump[12], to collect statistics about network (must be run with option -T). In the beginning, known as "training period", computer system is studying it's own behaviour. During this period, about 6 to 8 weeks, data from database are not reliable. Database updated continuously. By default, the update period is set to 5 minutes. System collects information about averages and variances describing "normal" behaviour. Collected data represents time period of two last months. So, there is no final "normal state". Being based on a kind of 2 month long sliding window, "normal state" changes all the time. According to collected statistical data, system can classify current state as a grade of deviation above / about the everage. Data collected by cfenvd are used by cfEngine to control system state[33]. Following data are recorded by cfenvd daemon:

- number of users
- number of root processes
- number of non-root processes
- percentage disk full for root disk
- number of incoming and outgoing sockets for netbiosns, netbiosdgm (port 137), netbiosssn (port 138), irc (port 194), cfEngine (port 5308), nfsd (port 2049), smtp (port 25), www (port 80), wwws (port 443) ftp (port 21), ssh (port 22) and telnet (port 23).

Conditions change and deamon adapts to it all the time. It has some internal inertia and need some time to react on gradual changes. In this way, short anomaly will not apply a big change on the "normal state" of the system. Changes are applied to the "normal state" all the time. Values that are outside of 2 month sliding window are gradually deprecated. Use of database is optimized and very effective. cfEngines' database doesn't contain all the data for past two month. Every time cfenvd collects a new portion of information, it will be processed together with data from database and result will be stored back to the database again. Using this method, cfEngine can hold small size of database (only few Mbytes) and provide very fast access to it[34].

6.4 Data analysis

Data collected by cfenvd represents a statistical information for a one week, based on measurement period about 2 month. To analyze data, we need to extract them from cfEngines' database. This operation can be done by using utility named cfenvgraph. This program can connect to database, read data, do necessary calculations and dump results to the screen and to the hard disk. Resulting files is a plain text files containing 2-dimentional tables, named cfenv-average and cfenv-stddev. Program has also an option to dump all values into separate files. Cfenv-average contains the weighted average values of all the recorded data. Cfenv-stddev contains the square root of the weighted variances with respect to the averages. Dumped files can be plotted using standard tools like vvgraph, Grace[1] (which is descendant of ACE/gr, also known as Xmgr) or gnuplot[35] or even Microsoft™Excel™.

cfEngine has a set of classes that are connected to "normal" values provided by cfenvd. Using these classes, administrator can define how system should react on different grades of deviation from normal state[34]. Graphical presentation of data collected by cfenvd can give a lot of information about system state and systems behaviour. Administrator can easily detect peaks of activity for different components of the system (including information

[1] http://plasma-gate.weizmann.ac.il/Grace/

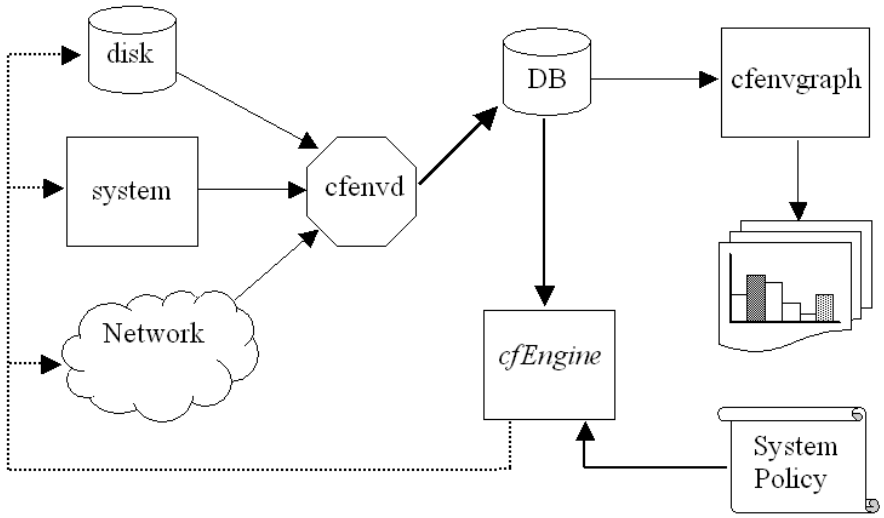

Figure 6.1: Data flow between system components and cfEngine components.

```
0 8.395726 54.091319 43.921564 63.017690 62.310175 0.000000
1 7.455108 49.427519 38.310022 58.946241 60.773019 0.000000 ...
2 8.349600 57.258946 21.456108 69.523942 15.808970 0.000000 ...
3 8.115748 50.400556 19.722852 61.140477 19.160703 0.000000 ...
4 8.391077 51.006726 35.058196 61.593354 20.329400 0.000000 ...
5 8.340509 50.554488 34.245553 61.605072 11.569993 0.000000 ...
...
106 8.512438 54.139680 44.087252 62.588476 38.766235 0.000000 ...
```

Figure 6.2: Table with data from *cfenv-average*.

```
1 0.513880 2.101109 2.822745 2.788337 43.506840 0.000000 ...
2 1.214265 7.964568 6.350651 9.594279 56.294593 0.000000 ...
3 1.378907 9.504458 4.511388 11.460011 10.495502 0.000000 ...
4 1.093983 6.154089 6.846431 7.544189 26.521101 0.000000 ...
5 1.006912 5.224509 4.363820 6.261866 20.302060 0.000000 ...
...
107 0.498419 1.890631 3.021648 2.478629 26.258488 0.000000 ...
```

Figure 6.3: Table with data from *cfenv-stddev*.

Figure 6.4: Example of graph generated by gnuplot[35] showing effective average threshold for normal amount of users in the system. It is easy to see that graph follows the week pattern.

Figure 6.5: Example of standard deviation error bars generated by gnuplot[35] showing weekly average of amount of users in the system.

about day of the week and time of the day). Having this information, administrator can tune system policy for best performance and most effective utilization of resources, it can help to choose the right time for critical tasks such as automated system backup[36]. We use data extracted by cfenvgraph for visualization in our prototype. It is not a real-time data, so we can concentrate us on presentation methods. In case of dynamical data, TimeSensor VRML node can be used to reload new refreshed World file repeatedly.
Internal logics of cfenvgraph operates with following data divided into several datasets:

- number of users
- number of root proceses
- number of other proceses
- data for the set of selected network sockets (incoming and outgoing)
- data for different network protocols (both incoming and outgoing)
- data for selected binaries

Each dataset has following data

- Maximum and Minimum values for all datasets
- Data for building Histograms
- Data for building SmoothHistograms
- Hurst exponent for selected datasets

First cfenvgraph finds MAX and MIN data in all datasets. The result of calculations is dumped to the screen including MIN, MAX and $\sqrt{(MAX)}$ values for all datasets. Then it calculates expected values for each dataset, filling the matrix in cfenv-average or dumping results into separate files. Data for histograms is also calculated. And finaly cfenvgraph estimates and print out Hurst Exponents for datasets. Network traffic is self-similar in many cases. The reasons for self-similarity is networks can be distribution of file sizes, dynamical nature of Ethernet or even interaction with human operator. Hurst exponent is a measure the degree of self-similarity and characteristics of long-range dependence. Lower value of Hurst exponent shows that there is no self-similarity in time serie (for example pure random processes have Hurst exponent about $0,5$). Values close to 1 shows long-range dependence. Hurst exponent is a scalar. It can not be calculated, but only estimated. Different estimaters can produce different results[37]AnomDetcfEnburgess-probabilistickaragiannis02longrange.

More information about cfEngine including source code and documentation can be obtained from the cfEngine web site[38].

Chapter 7

implementation

As a testbed for some concepts a system codenamed cfMagine was designed. The first prototype presented a scripting approach. There was designed a set of shell scripts that created a 3-dimentional scene in a cooperation with cfenvgraph and gnuplot[35]. Images generated by gnuplot were used as textures for simple VRML objects. This method had too much limitations and was not convenient to work with. Following one of the ideas from cfEngine we decided to avoid scripting and all well known probems that follows with it. cfEngine designed as a set of tools with small memory footprint, that rely most on it's own code and less on external shared libraries. Source code of cfEngine and all its' components is licensed under the GNU General Public License[1] and accessable from cfEngines' web site[38] and allowed for modifications. It was a natural choice to use it as a start point in developing prototype. Two prototypes was done. In the first one, new code was merged into the source code of cfenvgraph as much as possible. It was wrong. It was hard to handle and maintain the code. The last approach intercepts the original code of cfenvgraph only when it's really needed. VRML World generator is designed as the one of subroutines in cfenvgraph. Because of the nature of the data collected by cfEngine, it was choosen to concentrate us on testing and approving of different graph-placement metaphors, advantages and disadvantages of named models3. Charts used in prototype are just a simple curves or Bar charts build in XY-coordinate system. Each chart provides an image of a normal state for a one week period.

7.0.1 Objects

When Building graphs different methods were tested.

- PointSet node. In this approach we have tested simple node which is the set of points. PointSet contains color information, list over XYZ coordinates and can be easily generated. Building graph based on PointSet object is possible, but becase of small size of points, it will be not so easy to work with it.
- Simple line using node of type IndexedLineSet. IndexedLineSet is similar to PointSet. Points are now connected to each other creating polylines. Lines can only reflect light from other light-sources. It is impossible to apply texture on the line. There is no possibility to change the width of lines. This method was tested in earlier prototypes but was not good enough. Graph lines was very thin and not easy to recognize.
- Using Sphere nodes. Each point of the graph can be represented as a node of type Sphere with short radius. In comparison with PointSet node, this solution looks much better, but can not be used because of huge XML-overhead, big file-size and rendering difficulties.

[1] GNU GPL can be obtained from http://www.gnu.org/licenses/gpl.html

- Using Box node. Graphs made of lots of bars are huge and rendering of this objects goes really slow. This method was dropped.
- Using Cylinders and spheres. This method can create a graph i form of a beautiful pipe, but the size and amount of objects make it unreal to use it. It can be used for low resolution graphs.
- Elevation Grid was the node of our choice. Method creates a graph in a form of ribbon. The last prototype was build using this node type.

It was choosen to design graph objects based on self-glowing material (using emissiveColor). This helps to see objects best under any angle of view.

7.0.2 Overview

With series of Worlds for different periods of time and using Links forward and back along Timeline, operator can jump from Worlds for the past to the World of today. Each graph in the World can have links to the previous and the next graph. It is possible to point to the particular graph inside of the target World (using Anchor with reference to the ViewPoint directing against the right graph)5.3. Data from different hosts in network can be stacked together into the form of stacked BarChart or even into the form of surface[9]

7.0.3 Zoom and Filter

Tasks of Zooming and Filtering can be done in two steps. On the first step, computer filter information and make initial decisions about placement of objects. This will prepare better environment for operator on the step two. On the second step operator can manipulate objects if he or she find out that it's needed. Rearranging of objects can be automated with help of script that operator can select from the list. It can be also done manual using mouse pointer. Drag-and-drop function can be programmed using TouchSensors5.3.1. Some graphs are related to each other (for example SMTP and POP/IMAP). It can be a good idea to place them together on the same chart. Objects can be divided into several logically separated domains, such as domain of internal resources (memory, CPU, users) and domain of network resources. Graphs can be build using any scale. VRML can easily apply transformation on any object to fit the size to appropriate borders.

7.0.4 Details on Demand

When we need to show details related to diagrams, we can use different methods.

Extra information about particular graph can be provided in form of text document or HTML document. When user click on diagram, browser can load a new document containing data source and extra information related to the diagram. This can be done using VRML object $< Anchor >$ 5.3. All kinds of Sensors described in 5.3.1 can be used to provide Details on Demand. When user is far away from the chart, there is no reason to provide additional information. It will be unreadable anyway. The speed of Scene rendering is dependent on amount of objects and objects complexity. To increase rendering speed we can temporaly remove unsignificant objects and use Level of Details. When user comes close to the object, system activate additional information. Combining VisibilitySensors and ProximitySensors, system can provide different information depending on users position and distance to object. So we can use ideas from Zooming User Interface, among others Semantic Zooming2.1.6. Each part of the graph can provide information related to it's own nature. Information can even be redesigned using Interpolators**??**.

In case of cfEngine it's not so important to update scene often. Changes are small, normal state database is inertial and there is no need for real-time updates. Other systems

can benefit from updating information using a TimeSensor or Automated refresh based on use of cron deamon.

Importance of the graph in prototype is based on Hurst exponent measures6.4. A graph with smaller Hurst exponent has a rougher surface. A graph with larger Hurst exponent has smoother surface. cfEngine can be used to manage not only one single system, but also lots of systems. It can be good idea to use 3D to visualize all the collected information from several hosts in network. Because of limited time we were working on visualization model for one single host only.

Resulting VRML file containing Virtual World can be send to the administrator with e-mail or it can be published on the Web as well. These operations can be done on regular basis using cron deamon.

Chapter 8

Conclusions and Discussion

8.1 Methods

When we change visualization techniques from standard 2-dimentional presentation into the 3-dimentional, we can make good use of following methods and attributes. It is always an advantage to use different colors. Color change or intensity alteration can show changes on host state2.1.4.

8.2 Results

We approved that it is generally possible to go from traditional 2-dimentional presentations of data to the 3-dimentional. A person, changing environment from 2-dimentional to 3-dimentional can be confiused in the beginning. Effective manipulating with 3-dimentional scenes demands some experience and skills. User self can decide from what position graph can be viewed in a best possible way. It is not so important in a case of flat diagrams that we use in our prototype, but it can be really important for diagrams of other types for example in case of CityScape chart2.0.2. Navigation is one of the most important parts when one works with 3-dimentional information. Carefully predefined ViewPoints can help to avoid many problems. Some methods of ViewPoints locations were proposed as well. Possibility to ”feel” the user and to manipulate objects can make a new kind of interface that will optimize itself for better view. 3-dimentional visualization is not an universal solution, but a powerfull tool, when combined with operators skills and capability. VRML and X3D can be used to plot data and create users interfaces. We have tested different browsers and found out that only two of them show the best standard compliance. These are Xj3D from Web3D consortium and Octaga Player from Octaga AS (version for Microsoft Windows). Both are stable, provide great rendering performace and rich set of navigation tools. Xj3D has sensitive parcer and can be recommended to control the quality of code. Octaga is more liberal and can ignore some minor inaccuracy in syntax4.1. Traditional input devices such as keyboard and mouse are good suited for work with 3-dimentional presentations. Some browsers suport keyboard-shortcuts that can help operate faster. One can get better precision from use of Trackball or Tablet. Mouse is being managed by hand. Trackballs and tablets are managed by fingers, that gives more precision and accuracy. TouchScreen hardware was not tested because of lack of that type of hardware, but we can suppose that it will be usefull too. Multi-Touch interaction[1] looks really promising for 3-dimentional interactions. The nature of VRML files makes them attracktive as a medium of information interchange. XML-based files can be easily modified. If users will get an option to define their own ViewPoints focused on particular parts of diagrams, then this modified files can

[1] http://mrl.nyu.edu/ jhan/ftirtouch/

be exchanged with other persons in the same way as we exchange text documents today. Binary X3D format can make this process secure. Our prototype can be used as a part of cfEngine suite to visualize ”normal state” data.

8.3 Future works and improvements

The most promising for the next step can be investigation of possibilities for simultanous cooperation of several operators inside of the same World using Distributed interactive simulation (DIS) component.

8.4 Acknowledgements

Special thanks for Simen Hagen from Oslo University College for his support, comments and ideas.

Bibliography

[1] James Abello and Jeffrey Korn. MGV: A system for visualizing massive multidigraphs. IEEE Transactions on Visualization and Computer Graphics, 8(1):21–38, 2002.

[2] A. van Dam; D. Laidlaw; R. Simpson. Experiments in immersive virtual reality for scientific visualization, 2002.

[3] Tetsuji Takada; Hideki Koike. Tudumi: Information visualization system for monitoring and auditing computer logs.

[4] Ben Shneiderman Stuart K. Card, Jock Mackinlay. Readings in Information Visualization: Using Vision to Think.

[5] Kurt Cagle. Distributed user interfaces: Toward svg 1.2. In Chaomei Chen Vladimir Geroimenko, editor, Visualizing Information Using SVG and X3D, chapter 6, pages 119–153. Springer-Verlag London Limited, 2005.

[6] Mike Fisk; Steven A. Smith; Paul M. Weber; Satyam Kothapally; Thomas P. Caudell. Immersive network monitoring, 2003.

[7] John T. Stasko and Joseph F. Wehrli. Three-dimensional computation visualization. In Ephraim P. Glinert and Kai A. Olsen, editors, Proc. IEEE Symp. Visual Languages, VL, pages 100–107. IEEE Computer Society, 24–27 1993.

[8] Stephen G. Eick; Todd L. Graves; Alan F. Karr; Audris Mockus; Paul Schuster. Visualizing software changes. IEEE Transactions on Software Engineering, 28(4), April 2002.

[9] J. Brown and A. McGregor. Network performance visualization: Insight through animation, 2000.

[10] Tetsuji Takada; Hideki Koike. Mielog: A highly interactive visual log browser using information visualization and statistical analysis. In Proceedings of LISA 2002, November 2002.

[11] Iosif-Viorel Onut; Bin Zhu; Ali A. Ghorbani. Svision: A novel visual network-anomaly identification technique. Computers Security, Elsevier, October 2005.

[12] Tcpdump public repository. `http://www.tcpdump.org/`, 16 December 2005. last access.

[13] Kulsoom Abdullah; Chris Lee; Gregory Conti; John A. Copeland; John Stasko. Ids rainstorm: Visualizing ids alarms. October 2005. more information on the first page in file.

[14] IEEE Symposium on Visual Languages and Human Centric Computing. Integrating a Zoomable User Interfaces Concept into a Visual Language Meta-tool Environment, 2004.

[15] Eighth International Conference on Information Visualisation. ORRIL: a simple building blocks approach to zoomable user interfaces, 2004.

[16] Automatic construction of dynamic 3d metaphoric worlds: An application to network management.

[17] Kenneth C. Cox, Stephen G. Eick, and Taosong He. 3d geographic network displays. SIGMOD Record, 25(4):50–54, 1996.

[18] The VRML Consortium Incorporated. the Virtual Reality Modeling Language (VRML). ISO recommendation. International Organization for Standardization, 1997.

[19] Inc. Web3D Consortium. Extensible 3D (X3D) encodings. ISO recommendation. International Organization for Standardization, 2004 - 2006.

[20] An object-oriented 3D graphics toolkit. Proceedings of the 19th annual conference on Computer graphics and interactive techniques, 1992.

[21] A virtual information desk on the Internet, volume 1. The Third Russian-Korean International Symposium on Science and Technology, 1999. KORUS '99, 1999.

[22] D.R. Nadeau. Building virtual worlds with vrml. 19:18–29, 1999.

[23] Nicholas F. Polys. Publishing paradigms for x3d. In Chaomei Chen Vladimir Geroimenko, editor, Visualizing Information Using SVG and X3D, chapter 7, pages 153–181. Springer-Verlag London Limited, 2005.

[24] Don Brutzman. X3d-edit authoring tool for extensible 3d (x3d) graphics. In Chaomei Chen Vladimir Geroimenko, editor, Visualizing Information Using SVG and X3D, chapter 14, pages 285–292. Springer-Verlag London Limited, 2005.

[25] Zhigang Wen Le Jin. Adorning vrml worlds with environmental aspects. 21:6–9, 2001.

[26] Bob Crispen. Vrml post-production: The secret of the best vrml worlds on the web. InterActivity Magazine, July 1998.

[27] Ph.D. Anita D'Amico and Mark Larkin. Methods of visualizing temporal patterns in and mission impact of computer security breaches.

[28] M. Burgess and R. Ralston. Strategies for distributed resource administration using cfengine. Software-Practice and Experience, 1997.

[29] M. Burgess. Computer immunology. In Proceedings of the 12th System Administration Conference (USENIX/LISA), 1998.

[30] M. Burgess. Cfengine: a system configuration engine. Technical report, University of Oslo, 1993.

[31] M. Burgess. Automated system administration with feedback regulation. Software-Practice and Experience, 1998.

[32] Mark Burgess. Anomaly detection with cfenvd and cfenvgraph. Faculty of Engineering, Oslo University College, 2.1.18 edition, August 2005.

[33] M. Burgess. Recent developments in cfengine. In Unix.nl conference, Waardenburg; Netherlands, 2001.

[34] M. Burgess. Two dimensional time-series for anomaly detection and regulation in adaptive systems. In Proceeding of the IFIP/IEEE DSOM conference, 2002.

[35] GNUplot homepage. http://www.gnuplot.info/, May 2006. last access.

[36] M. Burgess. A tiny overview of cfengine: Convergent maintenance agent. In Proceedings of the 1st International Workshop on Multi-Agent and Robotic Systems, MARS/ICINCO, 2005.

[37] T. Karagiannis, M. Faloutsos, and M. Molle. A user-friendly self-similarity analysis tool, 2003.

[38] M. Burgess. The cfEngine web site. http://www.cfengine.org/, Mai 2006. last access.