Fall 1-31-2002

# Attack visualization for intrusion detection system

Mohammad A. Rabie

*New Jersey Institute of Technology*

# ABSTRACT

## ATTACK VISUALIZATION FOR INTRUSION DETECTION SYSTEM

by

**Mohammad A. Rabie**

Attacks detection and visualization is the process of attempting to identify instances of network misuse by comparing current activity against the expected actions of an intruder. Most current approaches to attack detection involve the use of rule-based expert systems to identify indications of known attacks. However, these techniques are less successful in identifying attacks, which vary from expected patterns. Artificial neural networks provide the potential to identify and classify network activity based on limited, incomplete, and nonlinear data sources. Presenting an approach to the process of Attack visualization that utilizes the analytical strengths of neural networks, and providing the results from a preliminary analysis of the network parameters being watched like Internet Protocol (IP) packet length, packet traffic, IP byte traffic, IP packet rate, IP byte rate, User Datagram Protocol (UDP) packet length, UDP packet traffic, UDP byte traffic, UDP packet rate, UDP byte rate, Heart Beat (HB) End-to-end delay, and HB Packet loss rate. Beside collected attack data, numerical simulated data was generated using the neural network sigmoids with Matlab. The characteristics of the obtained data showed lots of similarities with the actual collected network data. Further work is continuing to obtain different attack data using the Opnet simulating program.

# ATTACK VISUALIZATION FOR INTRUSION DETECTION SYSTEM

by
Mohammad A. Rabie

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

January 2002

# APPROVAL PAGE

## ATTACK VISUALIZATION FOR INTRUSION DETECTION SYSTEM

**Mohammad Rabie**

Dr. Constantine N. Manikopoulos, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Yun-Qing Shi, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. George E. Antoniou, Committee Member — Date
Professor of Computer Science, MSU

# BIOGRAPHICAL SKETCH

Author: Mohammad A. Rabie

Degree: Master of Science

Date: January 2002

**Undergraduate and Graduate Education:**

- Master of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ 2002

- Bachelor of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ 2000

- Bachelor of Science in Mathematics
  Bethlehem University, Bethlehem, Israel 1988

**Major:** Computer Engineering

To my beloved family, friends and teachers

# ACKNOWLEDGEMENT

I would like to express my deepest appreciation to Doctor Constantine Manikopoulos the chair of my committee and my advisor who not only served as my research supervisor, providing valuable insight and intuition but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Doctor Yun-Qing Shi from the department of electrical and computer engineering (ECE/NJIT) and Doctor George Antoniou of Montclair State University (MSU), for actively participating in my committee. I would like to extend my acknowledgement to Doctor Jay Jorgenson from City College for providing software tools and for his help in the simulations.

Many of my fellow graduate students in the Computer Networking laboratory and my friends at New Jersey Institute of Technology deserve recognition for their valuable support.

## TABLE OF CONTENTS

Chapter Page

# TABLE OF CONTENTS
## (Continued)

**Chapter** **Page**

# LIST OF TABLES

**Table** **Page**

# LIST OF FIGURES

Figure Page

**Figure** **Page**

# CHAPTER 1

# INTRODUCTION

Because of the increasing dependence, which companies and government agencies have on their computer networks; the importance of protecting these systems from attack is critical. A single intrusion of a computer network can result in the loss or unauthorized utilization or modification of large amounts of data and cause users to question the reliability of all of the information on the network. There are numerous methods of responding to a network intrusion, but they all require the accurate and timely identification of the attack. This paper presents an analysis of the network data that will make it easier determine the applicability of neural networks in the identification of instances of external attacks against a network. The results of tests conducted on a C++ programs and on another program called Parallax. Finally, the areas of future research that are being conducted in this area are discussed.

## 1.1 Overview

The timely and accurate detection of computer and network system intrusions has always been an elusive goal for system administrators and information security researchers. The individual creativity of attackers, the wide range of computer hardware and operating systems, and the ever- changing nature of the overall threat to target systems have contributed to the difficulty in effectively identifying intrusions. While the complexities of host computers already made intrusion detection a difficult endeavor, the increasing prevalence of distributed network-based systems and insecure networks such as the Internet has greatly increased the need for intrusion detection [20]. There are two general categories of attacks, which intrusion detection technologies attempt to identify - anomaly detection and misuse detection [1,13]. Anomaly detection identifies activities that vary from established patterns for users, or groups of users. Anomaly detection typically involves the creation of knowledge bases that contain the profiles of the monitored activities. The second general approach to intrusion detection is misuse.

This technique involves the comparison of a user's activities with the known behaviors of attackers attempting to penetrate a system [17,18]. While anomaly detection typically utilizes threshold monitoring to indicate when a certain established metric has been reached, misuse detection techniques frequently utilize a rule-based approach. When applied to misuse detection, the rules become scenarios for network attacks. The intrusion detection mechanism identifies a potential attack if a user's activities are found to be consistent with the established rules. The use of comprehensive rules is critical in the application of expert systems for intrusion detection it requires extensive analysis of network parameters, in order to produce a distribution for that attack on those parameters. Since another distribution may also be generated for the normal packets on those same parameters, then it would be correct to assume that all other packets, which don't follow the characteristics of the normal or attack traffic as an undefined region for that attack.

In this project I will consider two types of attacks, however, actual network data is only available for one type of attack (UDP). In order to produce more variety of attacks which carry different characteristics from the UDP attack packets. I used a data produced by running neural network sigmoid coded in VC++ (hyperbolic functions).

## 1.2 Network Data Parameters

Developing a data collection methodology requires an evaluation of the cost and benefit of the data in the particular domain of study. Our goal is to evaluate the effects of certain sampling parameters on the integrity and the cost of the resulting samples. In general a larger sample can more closely reflect the true parent population, but each instant of sampling imposes a cost in terms of CPU time, buffer space, and sampling interval, or amount of calendar time one can devote to deriving a particular estimate. The sampling frequency must therefore be weighed against the accuracy requirements and complexity of a given object.

The twelve-hour data collections for my experiment represent a brief interval, indeed itself a sample from the ongoing population of network traffic. For the purpose of my study I will treat the data collected as the true parent population, and the subpopulation drawn by our various sampling techniques as the samples. Standard statistical formulas generally rely on estimates of parameters of the parent population for the default case where the parent population is not known. Because the actual parameters of this parent population, we use them rather than estimate of them. Our goal is then to assess how close each sample is to its parent population for several key measurements. For each class, one can implement, or approximate, any particular method via timer-based. That is, one can use timers to trigger the selection of a packet for inclusion in a sample. Implementing this method at a variety of granularities allows a range of sampling fractions. Furthermore one can vary the interval over which one samples: 15 minutes, an hour, a day, etc. Since the progress are not time-homogeneous, it is not clear that spreading the same number of samples over a longer intervals will generate the same results.

Timer-driven sampling methods use a timer rather than a packet counter to trigger the selection of packets to include in the sample when the timer expires, we select the next packet to arrive. This is a necessary approximation but seemingly inconsequential. For population with a linear trend, uniform random sampling will be more efficient than symmetric sampling.

With the Opnet simulating tool lots of the network parameters could be collected, but the most significant ones that we concentrated on and will enable the neural network decide a user behavior without changing the overall behavior are listed below in the table. UDP, ICMP, SYN attack and other attacks shares these collected parameters which makes the data scalable to any tools and project that might utilize it. Variables X1 through X12 represent the following network parameters:

**Table 1: Collected Network Parameters**

| Collected Parameter List | |
|---|---|
| Parameter1 | IP in packet length |
| Parameter2 | IP in packet traffic |
| Parameter3 | IP in byte traffic. |
| Parameter4. | IP in packet rate. |
| Parameter5 | IP in-byte rate. |
| Parameter6 | UDP in packet length. |
| Parameter7 | UDP in packet traffic. |
| Parameter8 | UDP in byte traffic. |
| Parameter9 | UDP in packet rate. |
| Parameter10 | UDP in byte rate. |
| Parameter11 | HB End to end delay. |
| Parameter12 | HB Packet loss rate. |

## 1.3 Neural Network

Most current approaches to the process of detecting intrusions utilize some form of rule-based analysis. Rule-Based analysis relies on sets of predefined rules that are provided by an administrator, automatically created by the system, or both. Expert systems are the most common form of rule-based intrusion detection approaches [8, 24]. The early intrusion detection research efforts realized the inefficiency of any approach that required a manual review of a system audit trail. While the information necessary to identify attacks was believed to be present within the voluminous audit data, an effective review of the material required the use of an automated system. The use of expert system techniques in intrusion detection mechanisms was a significant milestone in the development of effective and practical detection-based information security systems [1, 8, 19, 21, 24, and 28]. An expert system consists of a set of rules that encode the knowledge of a human "expert".

These rules are used by the system to make conclusions about the security-related data from the intrusion detection system. Expert systems permit the incorporation of an extensive amount of human experience into a computer application that then utilizes that knowledge to identify activities that match the defined characteristics of attack. Unfortunately, expert systems require frequent updates to remain current. While expert systems offer an enhanced ability to review audit data, the required updates may be ignored or performed infrequently by the administrator. At a minimum, this leads to an expert system with reduced capabilities. At worst, this lack of maintenance will degrade the security of the entire system by causing the system's users to be misled into believing that the system is secure, even as one of the key components becomes increasingly ineffective over time. Rule-based systems suffer from an inability to detect attacks scenarios that may occur over an extended period of time. While the individual instances of suspicious activity may be detected by the system, they may not be reported if they appear to occur in isolation. Intrusion scenarios in which multiple attackers operate in concert are also difficult for these methods to detect because they do not focus on the state transitions in an attack, but instead concentrate on the occurrence of individual elements. Any division of an attack either over time or among several seemingly unrelated attackers is difficult for these methods to detect. Rule-based systems also lack flexibility in the rule-to-audit record representation. Slight variations in an attack sequence can affect the activity-rule comparison to a degree that the intrusion is not detected by the intrusion detection mechanism. While increasing the level of abstraction of the rule-base does provide a partial solution to this weakness, it also reduces the granularity of the intrusion detection device. A number of non-expert system-based approaches to intrusion detection have been developed in the past several years [4, 5, 6, 9, 15, 25, and 26]. While many of these have shown substantial promise, expert systems remain the most commonly accepted approach to the detection of attacks.

An artificial neural network consists of a collection of processing elements that are highly interconnected and transform a set of inputs to a set of desired outputs. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. By modifying the connections between the nodes the network is able to adapt to the desired outputs [9, 12]. Unlike expert systems, which can provide the user with a definitive answer if the characteristics, which are reviewed exactly, match those, which have been coded in the rule base, a neural network conducts an analysis of the information and provides a probability estimate that the data matches the characteristics, which it has been trained to recognize. While the probability of a match determined by a neural network can be 100%, the accuracy of its decisions relies totally on the experience the system gains in analyzing examples of the stated problem. The neural network gains the experience initially by training the system to correctly identify pre- selected examples of the problem. The response of the neural network is reviewed and the configuration of the system is refined until the neural network's analysis of the training data reaches a satisfactory level. In addition to the initial training period, the neural network also gains experience over time as it conducts analyses on data related to the problem.

A limited amount of research has been conducted on the application of neural networks to detecting computer intrusions. Artificial neural networks offer the potential to resolve a number of the problems encountered by the other current approaches to intrusion detection. Artificial neural networks have been proposed as alternatives to the statistical analysis component of anomaly detection systems, [5, 6, 10, 23, and 26]. Statistical Analysis involves statistical comparison of current events to a predetermined set of baseline criteria. The technique is most often employed in the detection of deviations from typical behavior and determination of the similarly of events to those which are indicative of an attack [14]. Neural networks were specifically proposed to identify the typical characteristics of system users and identify statistically significant variations from the user's established behavior. Artificial neural networks have also been proposed for use in the detection of computer viruses. In [7] and [9] neural networks were proposed as statistical analysis approaches in the detection of viruses and malicious software in computer networks. The neural network architecture, which was selected for [9] was a self-organizing feature map, which uses a single layer of neurons to represent knowledge from a particular domain in the form of a geometrically organized feature map. The proposed network was designed to learn the characteristics of normal system activity and identify statistical variations from the norm that may be an indication of a virus.

While there is an increasing need for a system capable of accurately identifying instances of attacks on a network there is currently no applied alternative to rule-based intrusion detection systems. This method has been demonstrated to be relatively effective if the exact characteristics of the attack are known. However, network intrusions are constantly changing because of individual approaches taken by the attackers and regular changes in the software and hardware of the targeted systems. Because of the infinite variety of attacks and attackers even a dedicated effort to constantly update the rule base of an expert system can never hope to accurately identify the variety of intrusions. The constantly changing nature of network attacks requires a flexible defensive system that is capable of analyzing the enormous amount of network traffic in a manner, which is less structured than rule-based systems. A neural network-based attack detection system could potentially address many of the problems that are found in rule-based systems.

The first advantage in the utilization of a neural network in the detection of instances of attack would be the flexibility that the network would provide. A neural network would be capable of analyzing the data from the network, even if the data is incomplete or distorted. Similarly, the network would possess the ability to conduct an analysis with data in a non-linear fashion. Both of these characteristics are important in a networked environment where the information, which is received, is subject to the random failings of the system. Further, because some attacks may be conducted against the network in a coordinated assault by multiple attackers, the ability to process data from a number of sources in a non-linear fashion is especially important. The inherent speed of neural networks is another benefit of this approach. Because the protection of computing resources requires the timely identification of attacks, the processing speed of the neural network could enable intrusion responses to be conducted before irreparable damage occurs to the system. Because the output of a neural network is expressed in the form of a probability the neural network provides a predictive capability to the detection of instances of attack. A neural network-based attack detection system would identify the probability that a particular event, or series of events, was indicative of an attack against the system. As the neural network gains experience it will improve its ability to determine where these events are likely to occur in the attack process. This information could then be used to generate a series of events that should occur if this is in fact an intrusion attempt. By tracking the subsequent occurrence of these events the system would be capable of improving the analysis of the events and possibly conducting defensive measures before the attack is successful. However, the most important advantage of neural networks in attack detection is the ability of the neural network to "learn" the characteristics of attacks and identify instances that are unlike any which have been observed before by the network. A neural network might be trained to recognize known suspicious events with a high degree of accuracy. While this would be a very valuable ability, since attackers often emulate the "successes" of others, the network would also gain the ability to apply this knowledge to identify instances of attacks, which did not match the exact characteristics of previous intrusions. The probability of an attack against the system may be estimated and a potential threat flagged whenever the probability exceeds a specified threshold.

## 1.4 Visualization Tool

Each collected data record consists of four sets of 67 numbers. One collected vector is shown in table 2 below. Checking the similarities of the vector and trying to minimize its parameters. Using a matlab program that utilizes the wavelet compression tools and concentrations a 13 dimensional vector (shown in table 3) was produced while keeping the same characteristics of the original vectors with 804 dimensions shown in table2. With the new produced vectors I was able to use a visualization tools to study further the data behavior and contrast the normal from the attack data.

Two tools were utilized to monitor the data behavior over the network built
Parallax
Multi-dimensional graphics tool with a two-dimensional scatter plots graphics representation tool
Scatter3D
A tool capable of producing virtual three dimensional scatter plots
These tools are discussed extensively with screen shots in chapters 2 and 3 with screen shots.

## Table 2: Sample Opnet Data Record

| Request Incoming | | | Reply Outgoing | | | Request Incoming | | | Reply Outgoing | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 64 | 64 | 0 | 64 | 64 | 8 | 64 | 64 | 8 | 64 | 64 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.5625 | 15.625 | 781.25 | 1.5625 | 15.625 | 781.25 | 1.5625 | 15.625 | 781.25 | 1.5625 | 15.625 | 781.25 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0.90625 | 0.96875 | 0 | 0.90625 | 0.96875 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0625 | 0.03125 | 0 | 0.0625 | 0.03125 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.03125 | 0 | 0 | 0.03125 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.125 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0.875 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| : | : | : | : | : | : | : | : | : | : | : | : |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Packet Length** | **Packet Rate** | **Byte Rate** | **Packet Length** | **Packet Rate** | **Byte Rate** | **Packet Length** | **Packet Rate** | **Byte Rate** | **Packet Length** | **Packet Rate** | **Byte Rate** |

Table 3: Wavelet Output of Table 2

| Parameter Symbol | Vector Parameter | Value |
|---|---|---|
| X1 | IP Packet Length | 0.0950100 |
| X2 | IP Packet Traffic | 0.0959900 |
| X3 | IP Byte Traffic | 0.4325300 |
| X4 | IP Packet Rate | 0.5653500 |
| X5 | IP Byte Rate | 0.4568760 |
| X6 | UDP Packet Length | 0.5646750 |
| X7 | UDP Packet Traffic | 0.3346567 |
| X8 | UDP Byte Traffic | 0.3556760 |
| X9 | UDP in-Packet Rate | 0.5367600 |
| X10 | UDP Byte Rate | 0.4356760 |
| X11 | HB End-to-End Delay | 0.5337670 |
| X12 | HB Packet Loss Rate | 0.3437670 |
|  | Label | 1 |

# CHAPTER 2

# PARALLAX

## 2.1 Introduction

The software is based on Parallel Coordinates, which is a methodology for the unambiguous (i.e. no loss of information) visualization of multivariate data. The discovery of multivariate/multidimensional relations in a dataset is transformed into a 2-D pattern recognition problem. The software's unique interface, queries, and boolean operators enable the visual/interactive discovery of complex relations in multivariate datasets, and in turn finding the effect these relations have on various objectives. Unexpected relations have been discovered in datasets with many variables from which sensitivities, repetitive patterns, other trends and salient properties are found. The visualization not only helps the discovery process but also the presentation and explanation of the results.

## 2.2 Parallax Specification

### 2.2.1 Parallax Header Code

A header must be added to any parallax data file in order to facilitate the access. The header consists of two lines where the specification of how many parameters is fed to parallax. Also it has a data starting point indicator to indicate that from here on it will be pure data for parallax to start scanning. Below is an example of parallax header.

```
nvars = n
x1 x2 x3 x4 x5 x6 ... ... ... ... x(n-1)ids = class
undefined_data = MISSING
data =
```

From the top header we could see that the data file consists of n parameters where the last one is an integer label. The undefined_data = MISSING indicates that there is no other data from out side should be looked at. The data = is the starting point of the data.

### 2.2.2 Parallax Data Files

Without the header the Parallax data files look simple and easy to follow. Each data file consists of specific number of parameters with a label positioned at the last column. There is no limit for the number of columns in the data file beside the computer screen. Visualizing more than 30 parameters the user can't make up the produced graphs clearly.

Sample data file for our purpose with 12 parameters and four labels is shown in the next page. The number of the different labels at the last column indicates the different types of data to be manipulated.

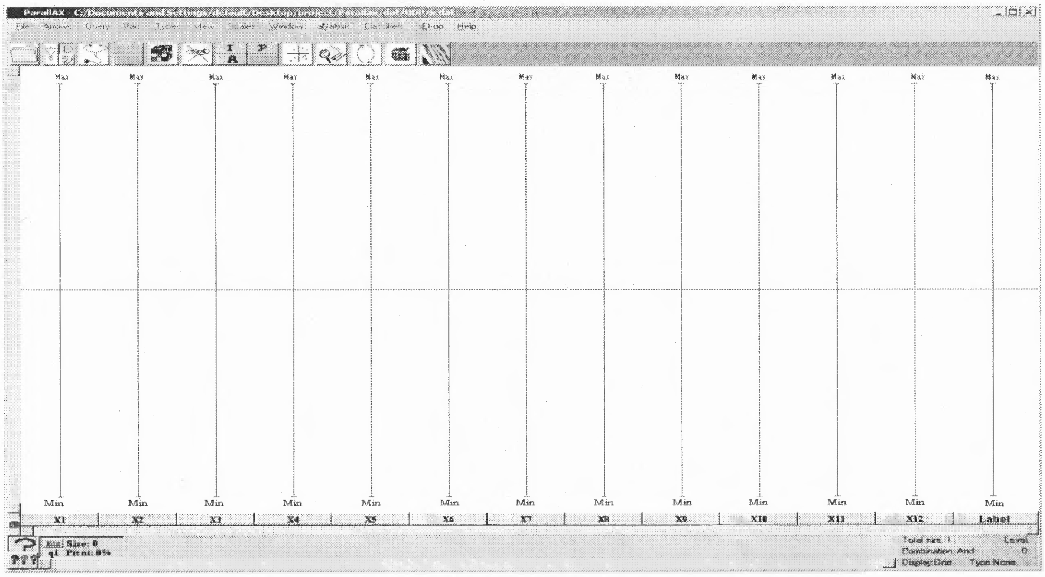
**Table 4: Sample Parallax Data File**

```
0.12536   0.940735  -0.586499 0.482003  0.389476  -0.214381 -0.600825 -0.269389 0.014457  -0.075249 0.991691  -0.023646 0 0 1 0
0.182065  0.179006  0.078896  0.013751  -0.064946 0.00702   0.147368  0.114794  -0.104759 -0.197704 0.000000  0.000000  0 1 0 0
-0.927622 -0.492614 -0.435281 -0.018856 -0.302513 0.045763  -0.71216  -0.031581 -0.651917 0.859393  0.379144  -0.818468 0 0 1 0
0.735239  0.738614  0.466809  0.035609  -0.143944 -0.05727  0.616653  0.550009  -0.09374  -0.49324  0.000000  0.000000  0 0 0 1
0.167862  0.167629  0.065629  0.076648  0.010366  -0.048653 0.116374  0.046142  0.000003  -0.129088 0.000000  0.000000  0 1 0 0
-0.608057 -0.391288 -0.620058 -0.447768 0.12178   -0.92553  -0.329081 0.974692  -0.860079 0.903345  0.289158  -0.862329 0 0 1 0
0.682182  0.690641  0.388895  0.174524  -0.111657 0.099757  0.755013  0.641853  -0.140615 -0.532956 0.000000  0.000000  0 0 0 1
0.751563  0.790538  0.356056  0.086959  -0.292073 -0.114288 0.610613  0.355514  -0.140615 -0.721245 0.000000  0.000000  0 0 0 1
0.56967   0.781351  0.285276  0.268981  0.012312  -0.354432 0.388265  0.220759  -0.0407   -0.583184 0.000000  0.000000  0 0 0 1
0.590407  0.750068  0.198551  0.100932  -0.200989 -0.198893 0.559177  0.230249  -0.09374  -0.530989 0.000000  0.000000  0 0 0 1
0.376057  0.496528  0.191715  0.183532  -0.158317 -0.160906 0.270261  0.177623  -0.105386 -0.438155 0.000000  0.000000  1 0 0 0
0.081109  0.122444  0.009061  -0.021171 -0.039815 -0.12142  0.085855  0.011036  -0.010144 -0.133474 0.000000  0.000000  0 1 0 0
0.179323  0.328548  -0.021295 0.112477  0.002998  -0.442612 0.17781   -0.124191 0.050454  -0.572747 0.000000  0.000000  0 0 0 1
0.386531  0.075796  -0.155725 0.128416  0.078614  -0.569842 -0.050465 -0.225071 0.019204  -0.461186 0.000000  0.000000  0 0 0 1
0.309781  0.210559  0.00595   -0.032536 -0.241038 -0.26232  0.034129  -0.057818 -0.105508 -0.454726 0.000000  0.000000  1 0 0 0
0.082304  0.082576  0.000513  -0.027207 -0.067814 -0.103449 0.027335  -0.019961 -0.065219 -0.165289 0.000000  0.000000  0 1 0 0
-0.991067 -0.888219 -0.445267 -0.08541  -0.819128 -0.738939 -0.239291 0.357064  -0.001649 0.251235  0.412362  0.955205  0 0 1 0
0.090919  0.085773  -0.035956 0.041853  -0.031493 -0.138274 -0.000692 -0.14762  -0.038033 -0.155536 0.000000  0.000000  0 1 0 0
0.081649  0.093047  -0.035195 0.008614  -0.008551 -0.145783 -0.021524 -0.099471 0.009327  -0.132993 0.000000  0.000000  0 1 0 0
-0.60097  0.143413  -0.330393 0.524701  -0.928772 -0.803426 -0.468137 0.985284  0.244573  -0.968757 0.525605  0.260571  0 0 1 0
0.420931  0.469677  0.212298  0.11715   -0.098598 -0.006823 0.381653  0.270204  -0.116326 -0.37335  0.000000  0.000000  1 0 0 0
0.281995  0.244851  -0.006376 0.050594  -0.195977 -0.537557 0.064542  -0.195402 -0.043296 -0.431215 0.000000  0.000000  0 0 0 1
0.310098  0.457406  0.000188  0.109845  -0.094451 -0.25076  -0.063073 -0.322889 -0.04171  -0.366315 0.000000  0.000000  1 0 0 0
0.54784   -0.468842 0.310207  -0.683995 0.1469    0.733181  -0.988448 -0.908958 0.834848  -0.008057 -0.342280 -0.520760 0 0 1 0
0.540163  0.617948  0.038513  0.026018  -0.154888 -0.652474 -0.027767 -0.052547 -0.262046 -0.654288 0.000000  0.000000  0 0 0 1
0.427912  0.428198  0.234772  0.039455  -0.279615 -0.022898 0.353072  0.220889  -0.181573 -0.427578 0.000000  0.000000  1 0 0 0
0.103212  0.14878   0.03891   0.035844  -0.040444 -0.075112 0.080813  0.039821  -0.006249 -0.133391 0.000000  0.000000  0 1 0 0
0.562605  0.118813  -0.183079 0.031938  -0.685757 0.011515  -0.001012 0.60775   0.112086  0.018164  0.779682  -0.045400 0 0 1 0
0.385408  0.466384  0.263139  0.069314  -0.10698  -0.043016 0.185828  0.172342  -0.111107 -0.425987 0.000000  0.000000  1 0 0 0
0.39691   0.376252  0.105295  0.017306  -0.027869 -0.075836 0.35096   0.199342  -0.096869 -0.297959 0.000000  0.000000  1 0 0 0
0.102334  0.119473  0.001952  0.023315  0.008827  -0.113807 0.047604  -0.032456 -0.033497 -0.148634 0.000000  0.000000  0 1 0 0
0.09881   0.130975  0.038949  0.0052    -0.011108 -0.096139 0.049002  -0.012252 -0.038033 -0.1395   0.000000  0.000000  0 1 0 0
0.487029  0.578334  0.08768   0.106901  -0.255168 -0.500687 -0.02447  -0.283626 -0.246421 -0.745222 0.000000  0.000000  0 0 0 1
0.555523  0.600427  0.078044  0.118759  -0.138509 -0.503524 -0.110183 -0.427955 -0.118825 -0.617452 0.000000  0.000000  0 0 0 1
0.465634  0.411553  0.219123  0.042901  -0.207664 -0.020758 0.295762  0.19384   -0.193335 -0.518522 0.000000  0.000000  1 0 0 0
0.094038  0.153947  -0.04299  0.015493  -0.046104 -0.107888 0.023016  -0.148987 -0.049693 -0.123958 0.000000  0.000000  0 1 0 0
0.414363  0.459998  0.251744  0.138343  -0.152157 -0.049941 0.382364  0.151317  -0.188581 -0.468389 0.000000  0.000000  1 0 0 0
0.557419  0.588284  -0.08668  0.040227  0.007935  -0.380724 0.010697  -0.326897 -0.243825 -0.591418 0.000000  0.000000  0 0 0 1
0.412576  0.452861  0.219816  0.118006  -0.14758  -0.079215 0.430265  0.227613  -0.099481 -0.398665 0.000000  0.000000  1 0 0 0
0.141117  0.173636  0.06587   0.005434  0.030067  -0.086546 0.097282  -0.003021 -0.025716 -0.119102 0.000000  0.000000  0 1 0 0
-0.026157 -0.026035 0.085929  -0.138667 0.639037  0.525285  -0.659718 0.816642  0.131451  -0.779257 -0.226820 0.598686  0 0 1 0
-0.079593 0.847612  0.950622  0.743204  0.086928  0.253753  -0.72738  0.049621  0.942426  -0.965532 0.399022  0.703007  0 0 1 0
0.465607  0.45257   0.254309  0.026578  -0.114604 -0.13501  0.152002  0.010353  -0.052778 -0.364546 0.000000  0.000000  1 0 0 0
0.127902  0.142895  0.026935  -0.005206 0.018162  -0.074153 0.059634  -0.004665 -0.049048 -0.141119 0.000000  0.000000  0 1 0 0
-0.689164 -0.242561 -0.853132 -0.304955 0.137112  0.889099  0.851219  -0.42508  0.193689  0.858222  -0.169030 -0.393538 0 0 1 0
-0.520589 -0.203537 0.792211  -0.434218 -0.014264 -0.800611 -0.255864 -0.629503 0.267274  0.772999  0.091207  -0.698349 0 0 1 0
0.37509   0.44669   0.119447  0.047756  -0.009949 -0.192706 0.19912   -0.025247 -0.10398  -0.336252 0.000000  0.000000  1 0 0 0
0.12599   0.993466  0.670263  0.596107  0.200022  0.899441  -0.525932 -0.468842 -0.269029 0.323504  -0.635059 -0.865760 0 0 1 0
0.401441  0.453207  0.151094  0.048539  -0.034137 -0.090026 0.324235  0.103788  -0.123023 -0.371155 0.000000  0.000000  1 0 0 0
0.144696  0.181542  0.093714  0.08056   -0.027239 -0.056568 0.122759  0.067737  -0.035032 -0.17962  0.000000  0.000000  0 1 0 0
0.14653   0.199043  0.131991  0.037061  -0.042765 -0.072661 0.044547  0.034524  -0.014037 -0.140283 0.000000  0.000000  0 1 0 0
0.138905  0.188936  0.133958  0.096651  0.009277  -0.024042 0.15643   0.122349  -0.01168  -0.113704 0.000000  0.000000  0 1 0 0
```

## 2.3 Network Traffic Analysis with Parallax Software

Figure 1 illustrates Parallax window where 13 axes is drawn with the maximum value to the top and the minimum value to the bottom on each axe. Data get scanned from the data files and axes are drawn accordingly with the maximum value for each parameter positioned on the top of the specific axe and the minimum value positioned at the bottom of the same axe for the same parameter as shown in figure 2. as a last step Parallax will connect these point horizontally to create the records.

## Figure 1: Parallax Default Window



## Figure 2:Sample One Attack Data File

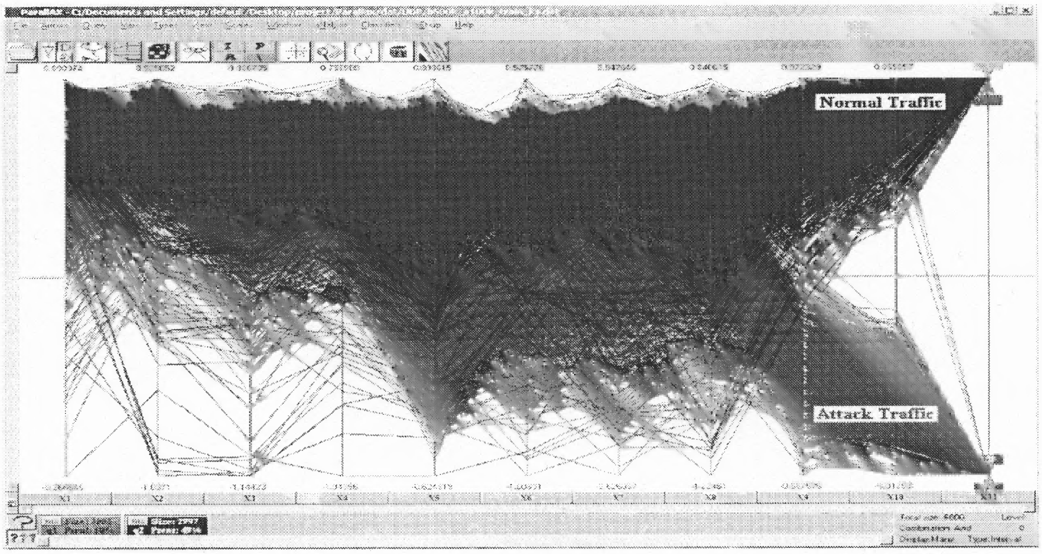
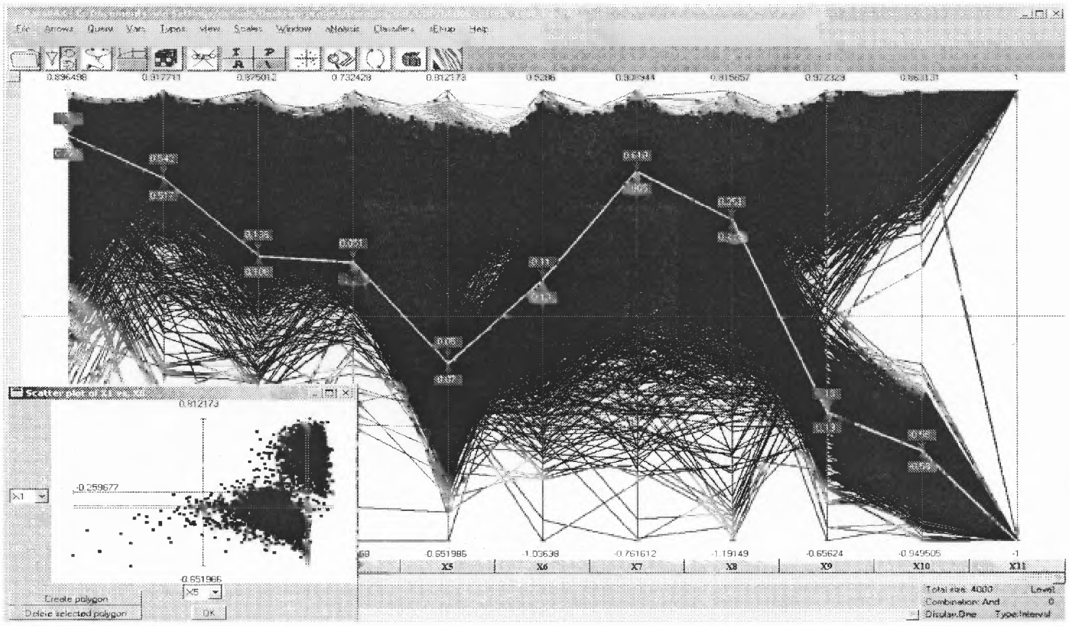


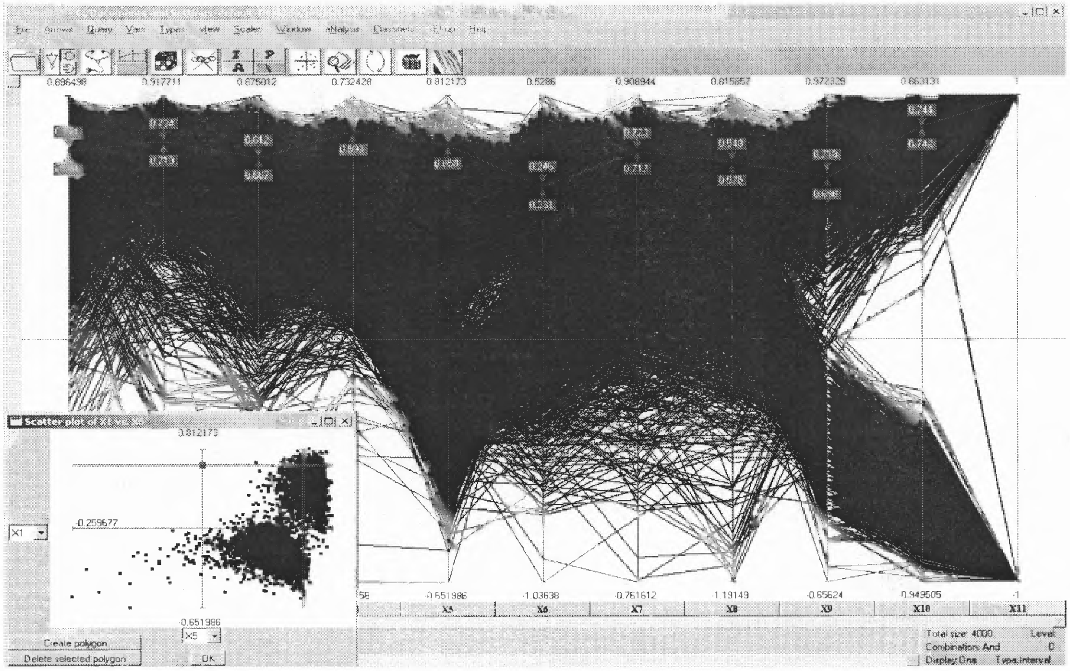
Figure 2 is a representation of one type of attack data file, where the normal UDP traffic is in blue and the UDP attack traffic in pink. The data records in a specific file could be visualized separately as shown in figure 3 and figure 4. Figure 5 illustrates one attack record and one normal record in the same window.

## Figure 3: Isolated One Attack Record



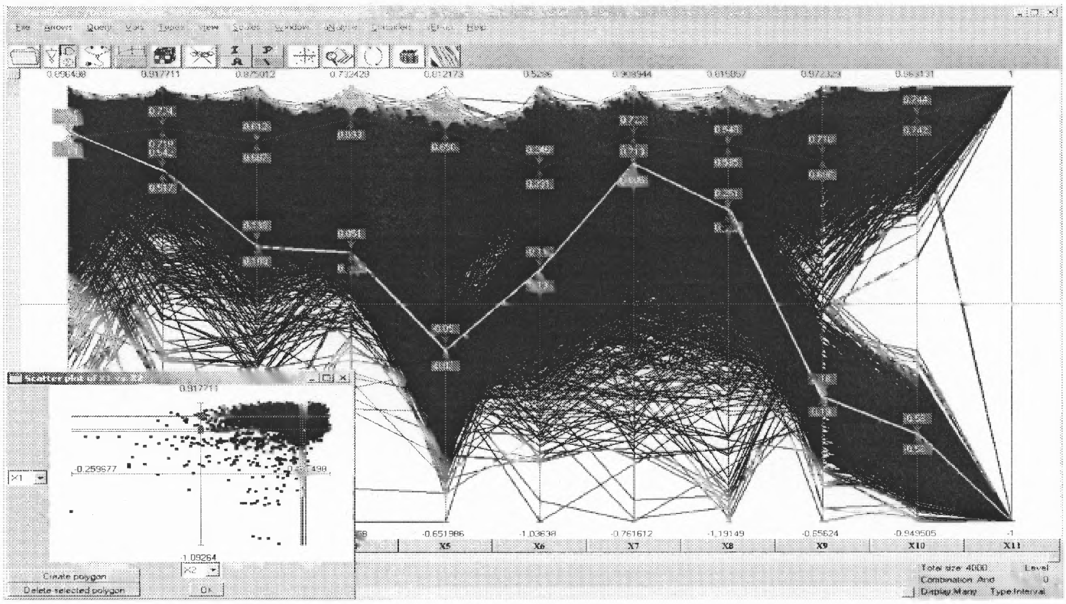
Total size: 4000 Level
Combination: And 0
Display:One Type:Interval

## Figure 4: Isolated Normal Data Record



Total size: 4000 Level
Combination: And 0
Display:One Type:Interval

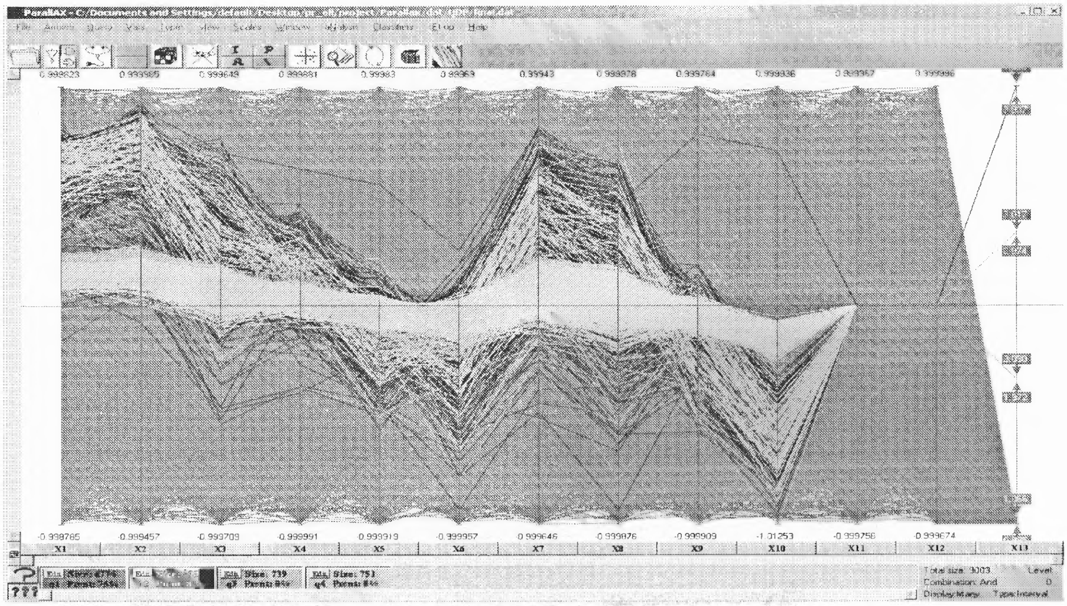
**Figure 5: Attack and Normal data in same Window**



### 2.3.1 Multiple Attack Data Files

Figure 6 is a representation of multiple type of attack data file where each color represents a different type of network attack. Each attack type could be isolated from the rest of the data and looked at separately.

**Figure 6: Multiple Attack Parallax Representation**



### 2.3.2 Parallax Scatter Plots

#### 2.3.2.1 Single Attack Scatter Plots

A comparison is made between two parameters like IP Packet Length with respect to IP Packet Traffic in the data sets collected from the network traffic. In this data, measurements were taken at the areas where the data concentrations lay most. Notice the following 9 figures (figure 7 - figure15) some parameters correlate with others in different ways. While at the time other parameters could be totally separable. The study of the parameters dependencies could be greatly reduced through out these scatter plots.

**Figure 7: Single Attack scatter plot (X1 vs. X2)**

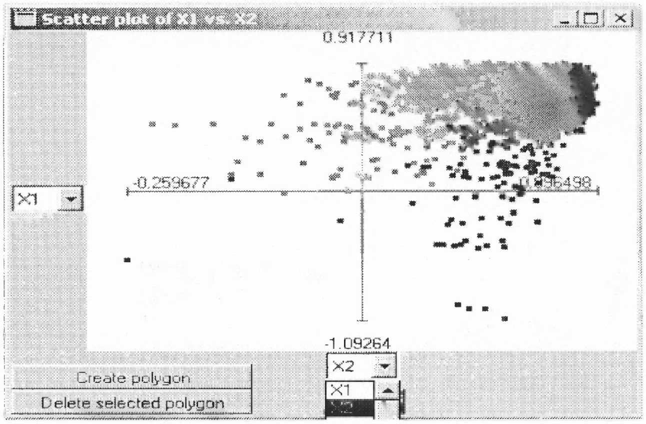


**Figure 8:Single Attack scatter plot (X1 vs. X3)**

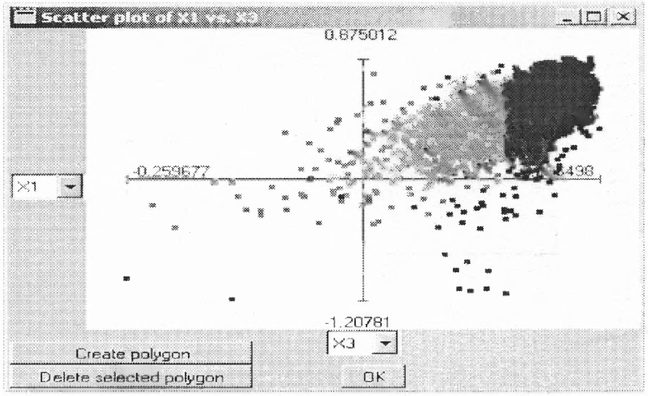


Figure 9: Single Attack scatter plot  (X1 vs. X7)

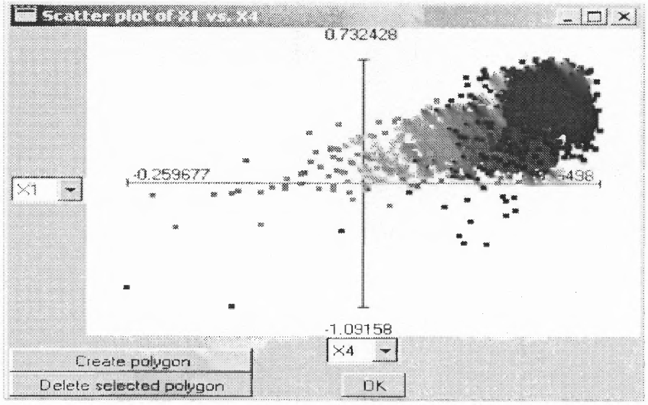


Figure 10: Single Attack scatter plot (X1 vs. X5)

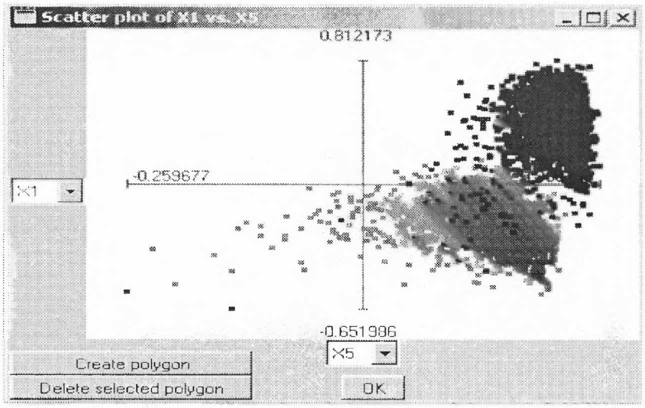


Figure 11: Single Attack scatter plot (X1 vs. X7)

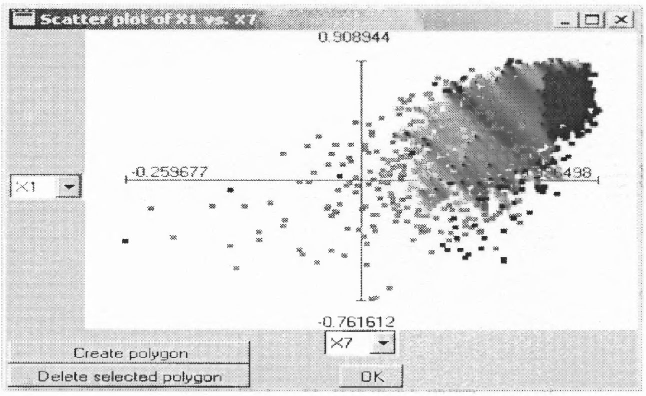


**Figure 12: Single Attack scatter plot (X1 vs. X8)**

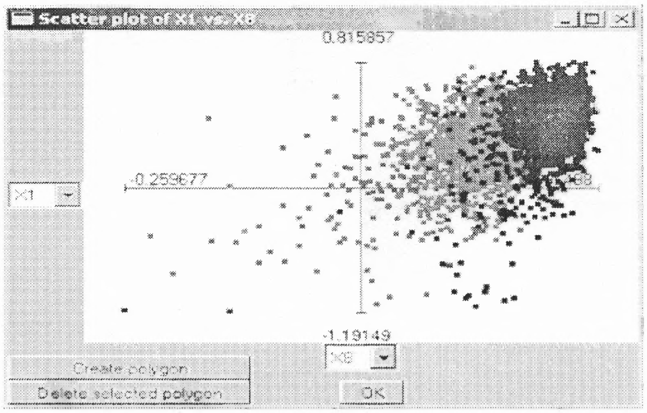


**Figure 13: Single Attack scatter plot (X1 vs. X9)**

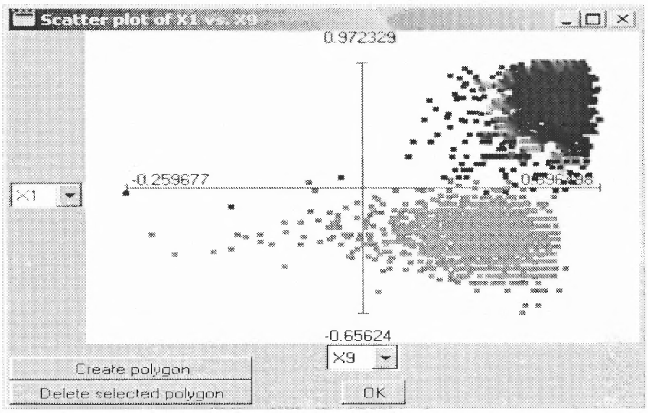


**Figure 14: Single Attack scatter plot (X1 vs. X10)**

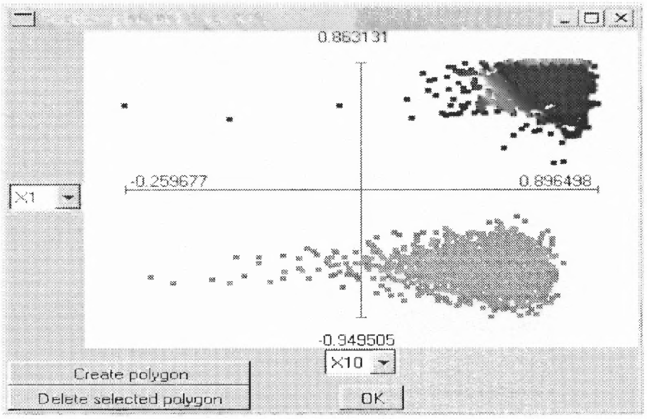
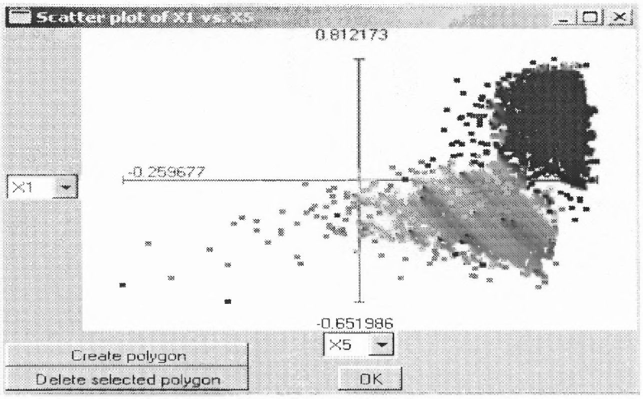


**Figure 15: Single Attack scatter plot (X1 vs. X5)**



### 2.3.2.2 Multiple Attack Scatter Plots

The need for multiple attack visualization is crucial when one more than one attack follow the same behaviour over the network or in the data set collected. At the same time it is very important to observe the various parameters that are very close in behaviour. Below is a set of scatter plots taken to represent the correlation of two parameters to the others in the same data flow or the same data set collected the network traffic.

**Figure 16: Multiple Attack scatter plot (X1 vs. X2)**

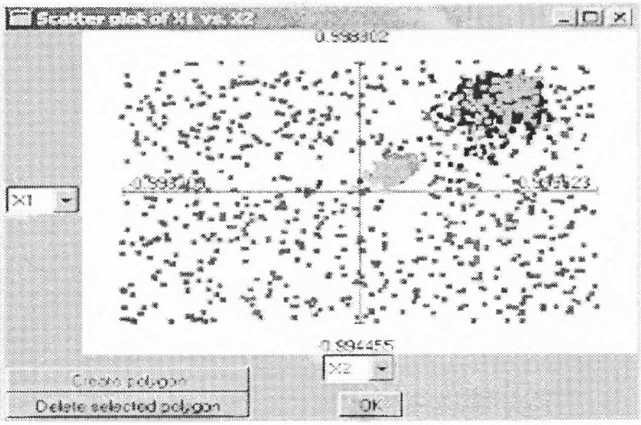


Figure 17: Multiple Attack scatter plot (X2 vs. X5)

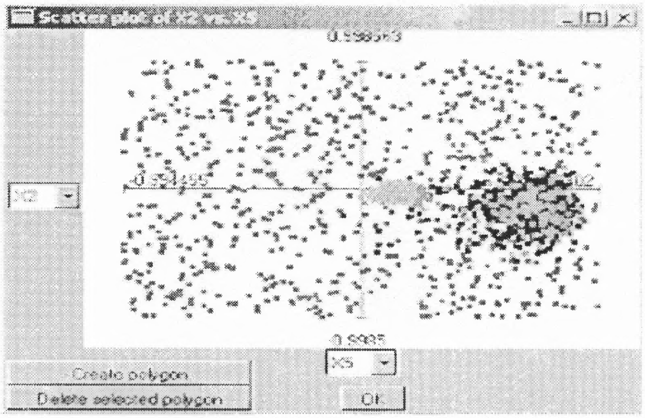


Figure 18: Multiple Attack scatter plot (X4 vs. X10)

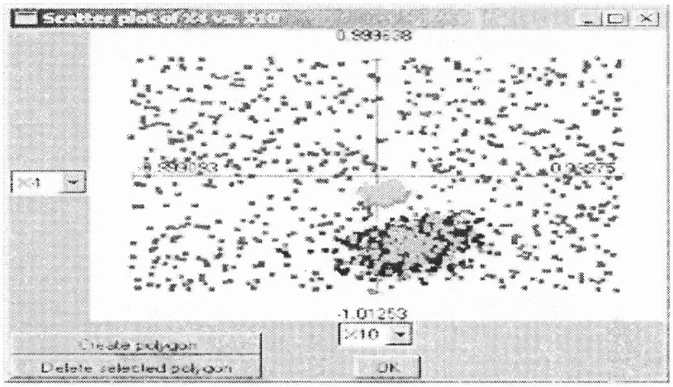


Figure 19: Multiple Attack scatter plot (X3 vs. X10)

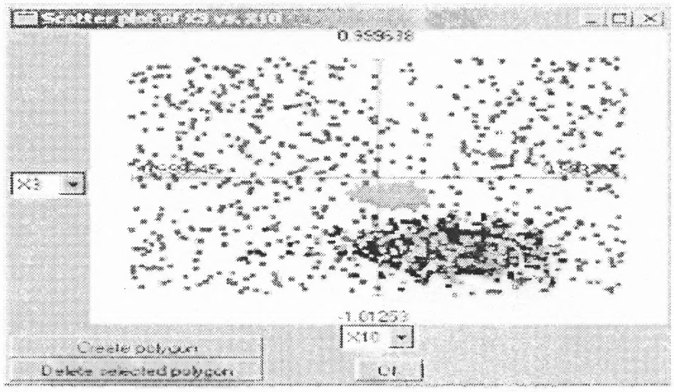


Figure 20: Multiple Attack scatter plot (X5 vs. X10)

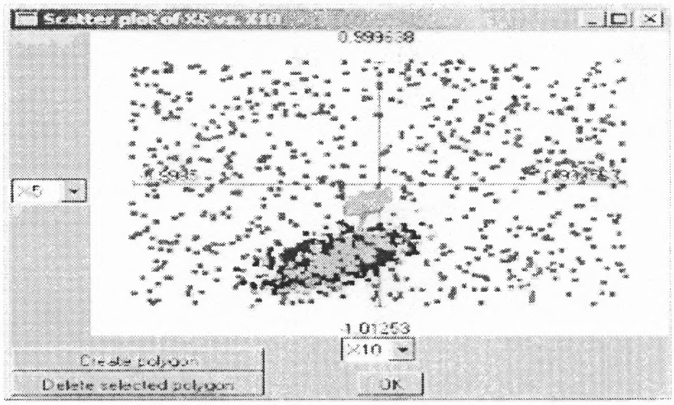


Figure 21: Multiple Attack scatter plot (X6 vs. X2)

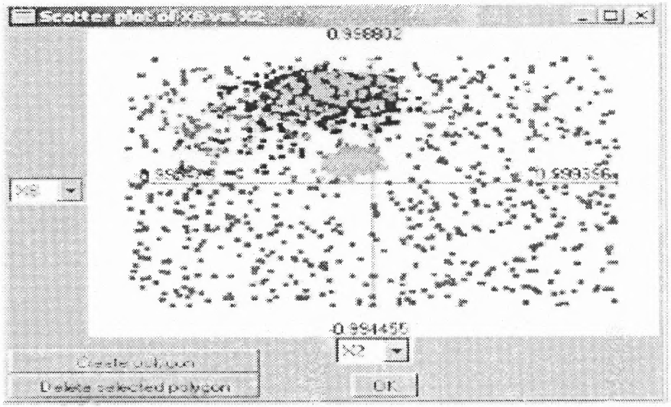


Figure 22: Multiple Attack scatter plot (X1 vs. X2)

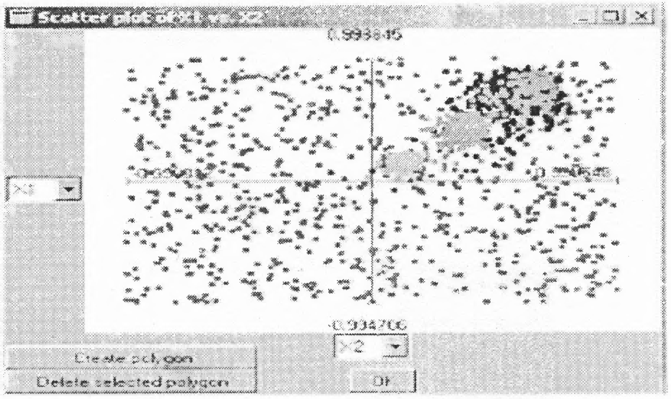


Figure 23: Multiple Attack scatter plot (X2 vs. X3)

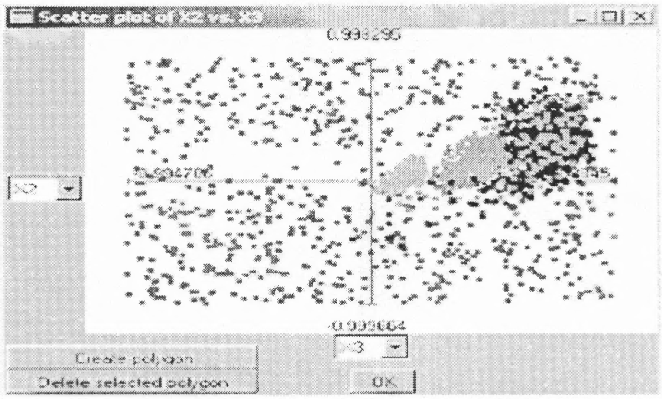


Figure 24: Multiple Attack scatter plot (X3 vs. X10)

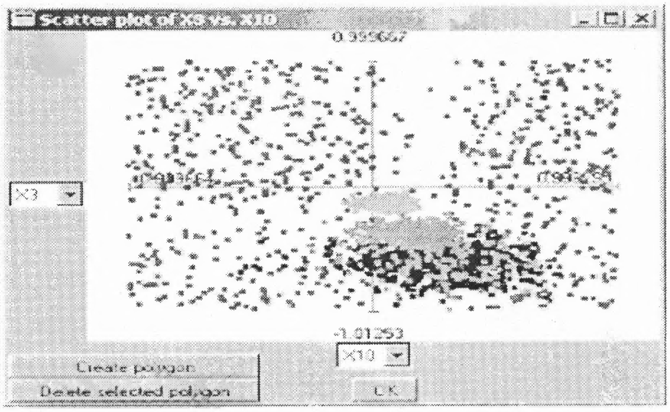


Figure 25: Multiple Attack scatter plot (X4 vs. X2)

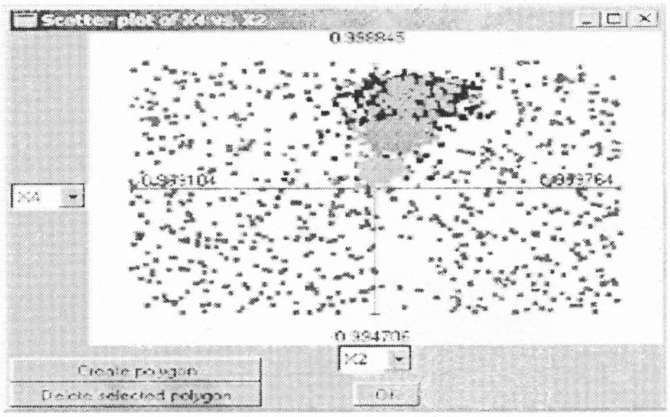


Figure 26: Multiple Attack scatter plot (X5 vs. X10)

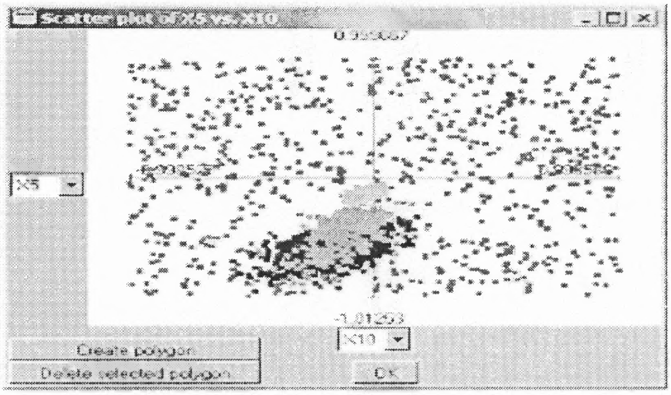


Figure 27: Multiple Attack scatter plot (X5 vs. X10)

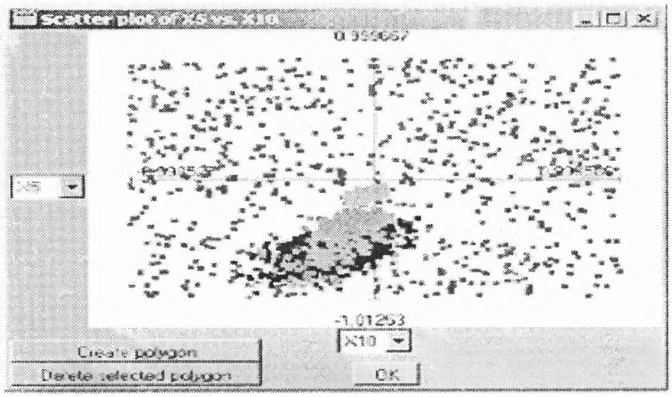


Figure 28: Multiple Attack scatter plot (X1 vs. X2)

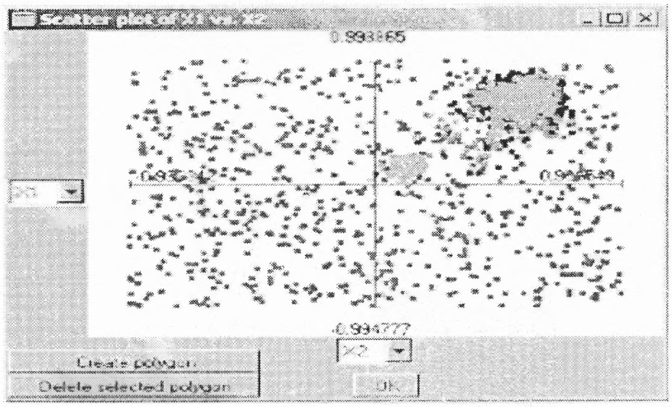


Figure 29: Multiple Attack scatter plot (X2 vs. X3)

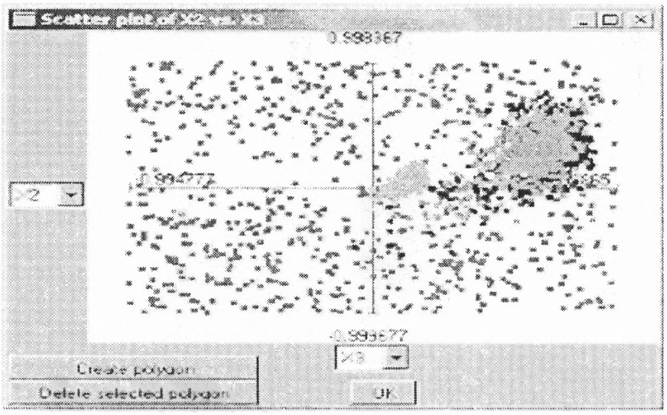
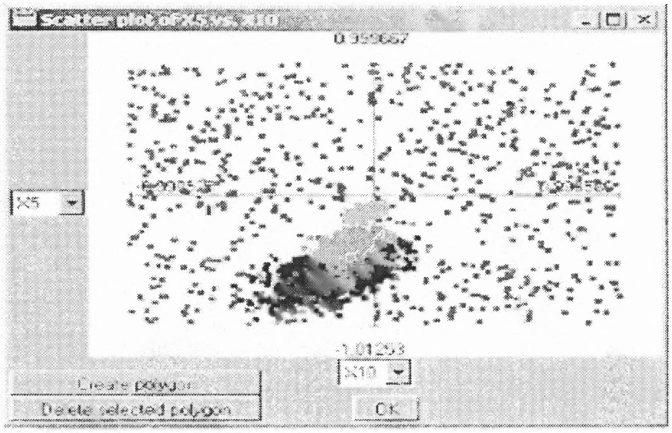


Figure 30: Multiple Attack scatter plot (X10 vs. X5)



## 2.4 Further Work

The preliminary results from observations give a positive indication of the potential offered by this approach, but a significant amount of research remains before it can function as an effective tool that give us a good understanding of the relations between the various network parameters. A complete system will require the ability to directly receive inputs from a network data stream and to visualize the data a three dimensional view to study the overlapping regions of the data types which will facilitate our judgment of the overlapping regions. The most difficult component of the analysis of network traffic by a neural network is the ability to effectively analyze the information in the data portion of an IP datagram. The various commands that are included in the data often provide the most critical element in the process of determining if an attack is occurring against a network. The most effective neural network architecture is also an issue that must be addressed. A feed- forward neural network that used a back propagation algorithm will be the best for its simplicity and reliability in a variety of applications. However, alternatives such as the self- organizing feature map also possess advantages in attack detection that may promote their use. In addition, an effective neural network-based approach to attack detection must be highly adaptive. Most neural network architectures must be retrained if the system is to be capable of improving its analysis in response to changes in the input patterns. Adaptive resonance theory ([2]) and self-organizing maps ([16]) offer an increased level of adaptability for neural networks, and these approaches are being investigated for possible use in an intrusion detection system. Finally, regardless of the initial implementation of a neural network-based intrusion detection system for attack detection it will be essential for the approach to be thoroughly tested in order to gain acceptance as a viable alternative to expert systems. Work has been conducted on taxonomies for testing intrusion detection systems ([3, 22]) that offer a standardized method of validating new technologies. Because of the questions that are certain to arise from the application of neural networks to intrusion detection, the use of these standardized methods is especially important.

# CHAPTER 3

# SCATTER3D

## 3.1 Introduction

A three-dimensional scatter plot is a trivariate plot in which a comparison of 3 measures is presented, one measure along each axis. You are then presented with a 3D cube in which the position of the current layer is indicated and a 2D plot in which the data for that layer is accurately represented. The 3D Scatter Plot can be animated with scatter3D giving you the ability to compare all of the layers effectively and quickly.

## 3.2 Scatter3D Specification

**Equation 1: Hex to Decimal example**

**hex 19** = $(1\text{x}16^{1}) + (9\text{x}16^{0}) = 25$

**Table 5: Hex to Decimal conversion table**

| **Second Digit**<br>**First Digit** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **1** | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| **2** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| **3** | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| **4** | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| **5** | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| **6** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| **7** | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| **8** | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| **9** | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| **A** | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| **B** | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| **C** | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| **D** | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| **E** | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| **F** | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |

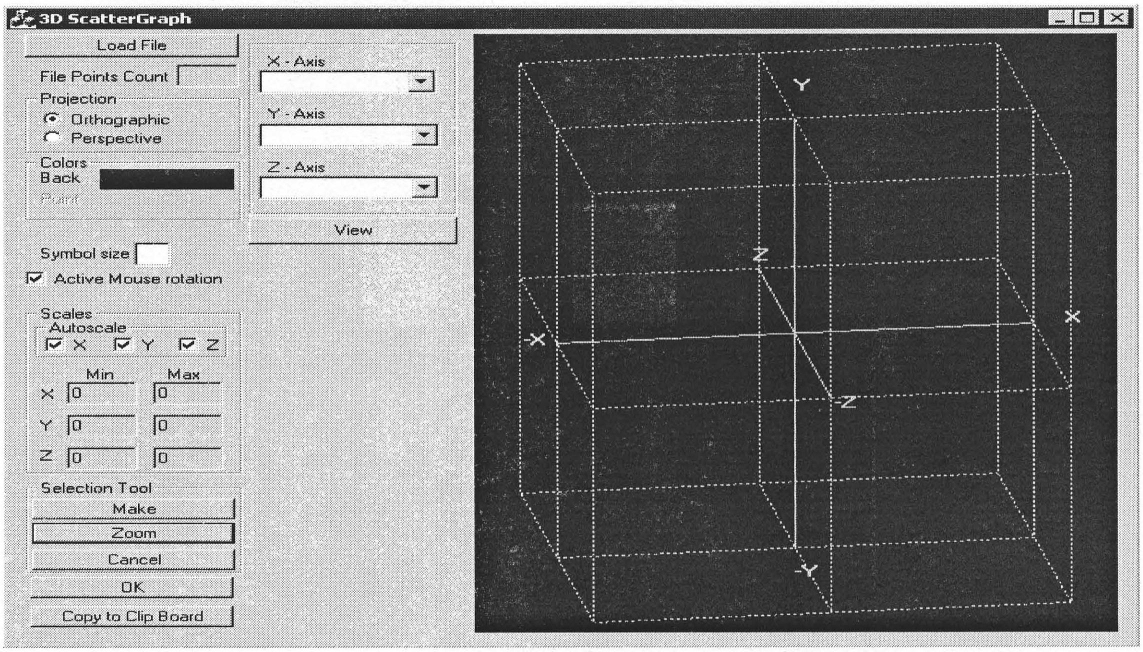
**Table 6: Color Coded Numbers (Red, Green, Blue)**

| Color | Name | HEX | DEC | Color | Name | HEX | DEC |
|---|---|---|---|---|---|---|---|
|  | Beige | F5F5DC | 245,245,220 | ▨ | Lightsteelblue | B0C4DE | 176,196,222 |
| ■ | Black | 0 | **0,0,0** | ▨ | Limegreen | 32CD32 | 50,205,50 |
|  | Blanchedalmo | FFEBCD | 255,235,205 | ■ | Magenta | FF00FF | **255,0,255** |
| ■ | Blue | 0000FF | **0,0,255** | ■ | Maroon | 800000 | 128,0,0 |
| ▨ | Darkkhaki | BDB76B | 189,183,107 |  | Oldlace | FDF5E6 | 253,245,230 |
| ■ | Darkolivegree | 556B2F | 85,107,47 | ▨ | Olive | 808000 | 128,128,0 |
| ▨ | Darkorange | FF8C00 | 255,140,0 | ▨ | Olive drab | 6B8E23 | 107,142,35 |
| ▨ | Dark orchid | 9932CC | 153,50,204 | ▨ | Orange | FFA500 | 255,165,0 |
| ▨ | Dark salmon | E9967A | 233,150,122 | ▨ | Orange red | FF4500 | 255,69,0 |
| ▨ | Darkseagreen | 8FBC8F | 143,188,143 | ▨ | Orchid | DA70D6 | 218,112,214 |
| ■ | Darkslateblue | 483D8B | 72,61,139 | ▨ | Pale goldenrod | EEE8AA | 238,232,170 |
| ■ | Darkslategray | 2F4F4F | 47,79,79 | ▨ | Pale green | 90EE90 | 144,238,144 |
| ▨ | Dark turquoise | 00CED1 | 0,206,209 | ▨ | Pale turquoise | AFEEEE | 175,238,238 |
| ■ | Dark violet | 9400D3 | 148,0,211 | ▨ | Palevioletred | DB7093 | 219,112,147 |
| ■ | Firebrick | B22222 | 178,34,34 | ▨ | Plum | DDA0DD | 221,160,221 |
| ■ | Forest green | 228B22 | 34,139,34 | ■ | Purple | 800080 | 128,0,128 |
| ▨ | Gainesboro | DCDCDC | 220,220,220 | ■ | Red | FF0000 | **255,0,0** |
|  | Ghostwrite | F8F8FF | 248,248,255 | ▨ | Rosy brown | BC8F8F | 188,143,143 |
| ▨ | Gold | FFD700 | 255,215,0 | ▨ | Royal blue | 41690 | 65,105,225 |
| ▨ | Goldenrod | DAA520 | 218,165,32 | ▨ | Salmon | FA8072 | 0,128,114 |
| ▨ | Gray | 7F7F7F | 127,127,127 | ▨ | Sandy brown | F4A460 | 244,164,96 |
| ■ | Green | 8000 | 000,128,000 | ▨ | Sea green | 2E8B57 | 46,139,87 |
| ▨ | Green yellow | ADFF2F | 173,255,47 |  | Seashell | FFF5EE | 255,245,238 |
| ▨ | Hotlink | FF69B4 | 255,105,180 | ▨ | Silver | C0C0C0 | 192,192,192 |
| ■ | Indian red | CD5C5C | 205,92,92 | ▨ | Sky-blue | 87CEEB | 135,206,235 |
| ■ | Indigo | 4B0082 | 75,0,130 | ▨ | Slate blue | 6A5ACD | 106,90,205 |
| ▨ | Lawn green | 7CFC00 | 124,252,0 | ▨ | Tan | D2B48C | 210,180,140 |
|  | Lemon chiffon | FFFACD | 255,0,205 | ■ | Teal | 8080 | 0,128,128 |
| ▨ | Light blue | ADD8E6 | 173,216,230 | ▨ | Thistle | D8BFD8 | 216,191,216 |
| ▨ | Light coral | F08080 | 240,128,128 | ▨ | Tomato | FF6347 | 255,99,71 |
| ▨ | Light cyan | E0FFFF | 224,255,255 | ▨ | Turquoise | 40E0D0 | 64,224,208 |
| ▨ | Lightered | D3D3D3 | 211,211,211 | ▨ | Wheat | F5DEB3 | 245,222,179 |
| ▨ | Light pink | FFB6C1 | 255,182,193 |  | White | FFFFFF | **255,255,255** |
| ▨ | Light salmon | FFA07A | 255,160,122 | ▨ | Yellow | FFFF00 | **255,255,0** |

## Table 8: Sample Scatter3D Input Data file

```
0.391402  0.379052  0.146744  0.218869  0.058243  -0.154056 0.136364  0.026027  -0.052056 -0.336626 0.000000  0.000000  32768
0.114726  0.177781  [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] -0.157757 0.000000  0.000000  16776960
-0.703927 0.080202  -0.137251 -0.061976 -0.535594 0.451147  -0.773159 0.411605  0.556766  -0.275018 -0.251367 -0.573110 16711680
0.107513  0.104682  -0.943591 0.575237  -0.553928 0.171986  -0.651007 -0.507398 -0.876832 0.430235  -0.257824 -0.821946 16711680
0.164872  0.192417  0.063283  0.029627  -0.01803  -0.003151 0.110798  0.042836  -0.058346 -0.111349 0.000000  0.000000  16776960
0.747117  0.809016  0.601406  0.130965  -0.2241   0.018744  0.378346  0.192083  -0.328115 -0.693127 0.000000  0.000000  16711935
-0.944184 0.290334  -0.955873 0.350699  -0.819614 0.323562  -0.656014 0.618302  0.441047  0.270474  -0.555522 -0.200908 16711680
0.187959  0.208043  0.114206  0.029943  -0.058109 0.021972  0.075604  0.003616  -0.09319  -0.174798 0.000000  0.000000  16776960
0.434053  0.41497   0.280114  0.138378  -0.111335 -0.083319 0.345815  0.270825  -0.083166 -0.349351 0.000000  0.000000  32768
0.010215  -0.171472 0.749882  -0.836213 0.582469  -0.764543 -0.985447 -0.33555  -0.135984 0.345018  0.304784  0.511508  16711680
-0.236394 -0.835222 0.641857  -0.589062 0.029886  0.150342  0.273245  0.547866  0.781407  0.282123  -0.076981 0.232167  16711680
0.334462  0.394018  0.228194  0.114896  -0.14569  -0.238689 0.07606   0.090446  -0.053336 -0.378295 0.000000  0.000000  32768
0.689185  0.770157  0.540454  0.313924  -0.250605 0.05282   0.552974  0.307618  -0.09374  -0.575341 0.000000  0.000000  16711935
0.179705  0.173493  0.164853  0.000955  -0.050681 -0.044768 0.111615  0.090536  -0.081597 -0.194046 0.000000  0.000000  16776960
0.697935  -0.129687 0.308189  0.89627   -0.174312 -0.505615 -0.566643 -0.952503 -0.254016 0.556431  -0.918911 -0.045830 16711680
0.12536   0.940735  -0.586499 0.482003  0.389476  -0.214381 -0.600825 -0.269389 0.014457  -0.075249 0.991691  -0.023646 16711680
0.182065  0.179006  0.078896  0.013751  -0.064946 0.00702   0.147368  0.114794  -0.104759 -0.197704 0.000000  0.000000  16776960
-0.927622 -0.492614 -0.435281 -0.018856 -0.302513 0.045763  -0.71216  -0.031581 -0.651917 0.859393  0.379144  -0.818468 16711680
0.735239  0.738614  0.466809  0.035609  -0.143944 -0.05727   0.616653  0.550009  -0.09374  -0.49324  0.000000  0.000000  16711935
0.167862  0.167629  0.065629  0.076648  0.010366  -0.048653 0.116374  0.046142  0.000003  -0.129088 0.000000  0.000000  16776960
-0.608057 -0.391288 -0.620058 -0.447768 0.12178   -0.92553  -0.329081 0.974692  -0.860079 0.903345  0.289158  -0.862329 16711680
0.682182  0.690641  0.388895  0.174524  -0.111657 0.099757  0.755013  0.641853  -0.140615 -0.532956 0.000000  0.000000  16711935
0.751563  0.790538  0.356056  0.086959  -0.292073 -0.114288 0.610613  0.355514  -0.140615 -0.721245 0.000000  0.000000  16711935
0.56967   0.781351  0.285276  0.268981  0.012312  -0.354432 0.388265  0.220759  -0.0407   -0.583184 0.000000  0.000000  16711935
0.590407  0.750068  0.198551  0.100932  -0.200989 -0.198893 0.559177  0.230249  -0.09374  -0.530989 0.000000  0.000000  16711935
0.376057  0.496528  0.191715  0.183532  -0.158317 -0.160906 0.270261  0.177623  -0.105386 -0.438155 0.000000  0.000000  32768
0.081109  0.122444  0.009061  -0.021171 -0.039815 -0.12142  0.085855  0.011036  -0.010144 -0.133474 0.000000  0.000000  16776960
0.179323  0.328548  -0.021295 0.112477  0.002998  -0.442612 0.17781   -0.124191 0.050454  -0.572747 0.000000  0.000000  16711935
0.386531  0.075796  -0.155725 0.128416  0.078614  -0.569842 -0.050465 -0.225071 0.019204  -0.461186 0.000000  0.000000  16711935
0.309781  0.210559  0.00595   -0.032536 -0.241038 -0.26232  0.034129  -0.057818 -0.105508 -0.454726 0.000000  0.000000  32768
0.082304  0.082576  0.000513  -0.027207 -0.067814 -0.103449 0.027335  -0.019961 -0.065219 -0.165289 0.000000  0.000000  16776960
-0.991067 -0.888219 -0.445267 -0.08541  -0.819128 -0.738939 -0.239291 0.357064  -0.001649 0.251235  0.412362  0.955205  16711680
0.090919  0.085773  -0.035956 0.041853  -0.031493 -0.138274 -0.000692 -0.14762  -0.038033 -0.155536 0.000000  0.000000  16776960
0.081649  0.093047  -0.035195 0.008614  -0.008551 -0.145783 -0.021524 -0.099471 0.009327  -0.132993 0.000000  0.000000  16776960
-0.60097  0.143413  -0.330393 0.524701  -0.928772 -0.803426 -0.468137 0.985284  0.244573  -0.968757 0.525605  0.260571  16711680
0.420931  0.469677  0.212298  0.11715   -0.098598 -0.006823 0.381653  0.270204  -0.116326 -0.37335  0.000000  0.000000  32768
0.281995  0.244851  -0.006376 0.050594  -0.195977 -0.537557 0.064542  -0.195402 -0.043296 -0.431215 0.000000  0.000000  16711935
0.310098  0.457406  0.000188  0.109845  -0.094451 -0.25076  -0.063073 -0.322889 -0.04171  -0.366315 0.000000  0.000000  32768
0.54784   -0.468842 0.310207  -0.683995 0.1469    0.733181  -0.988448 -0.908958 0.834848  -0.008057 -0.342280 -0.520760 16711680
0.540163  0.617948  0.038513  0.026018  -0.154888 -0.652474 -0.027767 -0.052547 -0.262046 -0.654288 0.000000  0.000000  16711935
0.427912  0.428198  0.234772  0.039455  -0.279615 -0.022898 0.353072  0.220889  -0.181573 -0.427578 0.000000  0.000000  32768
0.103212  0.14878   0.03891   0.035844  -0.040444 -0.075112 0.080813  0.039821  -0.006249 -0.133391 0.000000  0.000000  16776960
0.562605  0.118813  -0.183079 0.031938  -0.685757 0.011515  -0.001012 0.60775   0.112086  0.018164  0.779682  -0.045400 16711680
0.385408  0.466384  0.263139  0.069314  -0.10698  -0.043016 0.185828  0.172342  -0.111107 -0.425987 0.000000  0.000000  32768
0.39691   0.376252  0.105295  0.017306  -0.027869 -0.075836 0.35096   0.199342  -0.096869 -0.297959 0.000000  0.000000  32768
0.102334  0.119473  0.001952  0.023315  0.008827  -0.113807 0.047604  -0.032456 -0.033497 -0.148634 0.000000  0.000000  16776960
0.09881   0.130975  0.038949  0.0052    -0.011108 -0.096139 0.049002  -0.012252 -0.038033 -0.1395   0.000000  0.000000  16776960
0.487029  0.578334  0.08768   0.106901  -0.255168 -0.500687 -0.02447  -0.283626 -0.246421 -0.745222 0.000000  0.000000  16711935
0.555523  0.600427  0.078044  0.118759  -0.138509 -0.503524 -0.110183 -0.427955 -0.118825 -0.617452 0.000000  0.000000  16711935
0.465634  0.411553  0.219123  0.042901  -0.207664 -0.020758 0.295762  0.19384   -0.193335 -0.518522 0.000000  0.000000  32768
0.094038  0.153947  -0.04299  0.015493  -0.046104 -0.107888 0.023016  -0.148987 -0.049693 -0.123958 0.000000  0.000000  16776960
0.414363  0.459998  0.251744  0.138343  -0.152157 -0.049941 0.382364  0.151317  -0.188581 -0.468389 0.000000  0.000000  32768
0.557419  0.588284  -0.08668  0.040227  0.007935  -0.380724 0.010697  -0.326897 -0.243825 -0.591418 0.000000  0.000000  16711935
0.412576  0.452861  0.219816  0.118006  -0.14758  -0.079215 0.430265  0.227613  -0.099481 -0.398665 0.000000  0.000000  32768
0.141117  0.173636  0.06587   0.005434  0.030067  -0.086546 0.097282  -0.003021 -0.025716 -0.119102 0.000000  0.000000  16776960
-0.026157 -0.026035 0.085929  -0.138667 0.639037  0.525285  -0.659718 0.816642  0.131451  -0.779257 -0.226820 0.598686  16711680
-0.079593 0.847612  0.950622  0.743204  0.036928  0.253753  -0.72738  0.049621  0.942426  -0.965532 0.399022  0.703007  16711680
0.465607  0.45257   0.254309  0.026578  -0.114604 -0.13501  0.152002  0.010353  -0.052778 -0.364546 0.000000  0.000000  32768
0.127902  0.142895  0.026935  -0.005206 0.018162  -0.074153 0.059634  -0.004665 -0.049048 -0.141119 0.000000  0.000000  16776960
-0.689164 -0.242561 -0.853132 -0.304955 0.137112  0.889099  0.851219  -0.42508  0.193689  0.858222  -0.169030 -0.393538 16711680
-0.520589 -0.203537 0.792211  -0.434218 -0.014264 -0.800611 -0.255864 -0.629503 0.267274  0.772999  0.091207  -0.698349 16711680
0.37509   0.44669   0.119447  0.047756  -0.009949 -0.192706 0.19912   -0.025247 -0.10398  -0.336252 0.000000  0.000000  32768
0.12599   0.993466  0.670263  0.596107  0.200022  0.899441  -0.525932 -0.468842 -0.269029 0.323504  -0.635059 -0.865760 16711680
0.401441  0.453207  0.151094  0.048539  -0.034137 -0.090026 0.324235  0.103788  -0.123023 -0.371155 0.000000  0.000000  32768
0.144696  0.181542  0.093714  0.08056   -0.027239 -0.056568 0.122759  0.067737  -0.035032 -0.17962  0.000000  0.000000  16776960
0.14653   0.199043  0.131991  0.037061  -0.042765 -0.072661 0.044547  0.034524  -0.014037 -0.140283 0.000000  0.000000  16776960
0.138905  0.188936  0.133958  0.096651  0.009277  -0.024042 0.15643   0.122349  -0.01168  -0.113704 0.000000  0.000000  16776960
0.425831  0.362339  0.096931  0.222968  0.097841  -0.1279   0.124228  -0.062758 0.029045  -0.219458 0.000000  0.000000  32768
0.359363  0.301472  -0.083987 0.087141  -0.028701 -0.152177 -0.047024 -0.244507 -0.097573 -0.361448 0.000000  0.000000  32768
0.127272  0.184022  0.131233  0.063589  -0.03081  -0.055133 0.103727  0.062795  -0.023359 -0.169097 0.000000  0.000000  16776960
-0.841078 -0.263482 -0.06267  -0.494374 -0.125198 0.261257  -0.770347 -0.95336  0.193705  0.49256   0.194165  0.792521  16711680
0.372545  0.420292  0.143603  0.160709  -0.12799  -0.043394 0.313619  0.035678  -0.099481 -0.45799  0.000000  0.000000  32768
```

**Figure 31: Default Scatter3D Window**



File Points Count

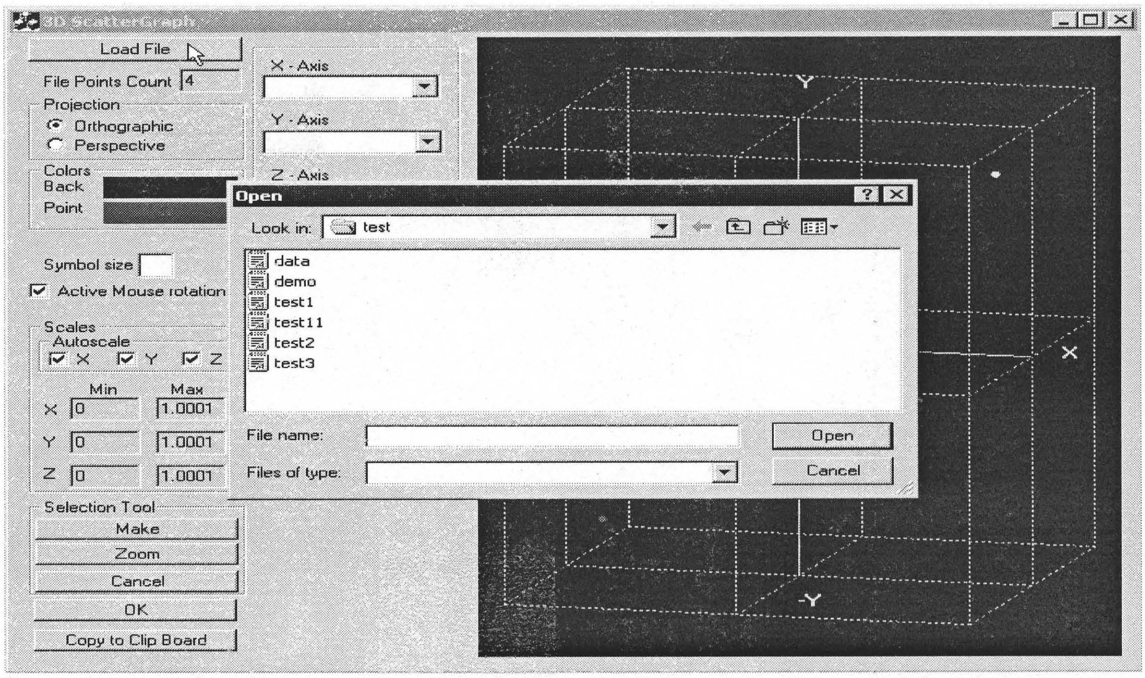
Projection
● Orthographic
○ Perspective

☑ Active Mouse rotation

Scales
Autoscale ☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

**Figure 32: Loading Files into Program**



File Points Count 4

Projection
● Orthographic
○ Perspective

☑ Active Mouse rotation

Scales
Autoscale ☑ X ☑ Y ☑ Z

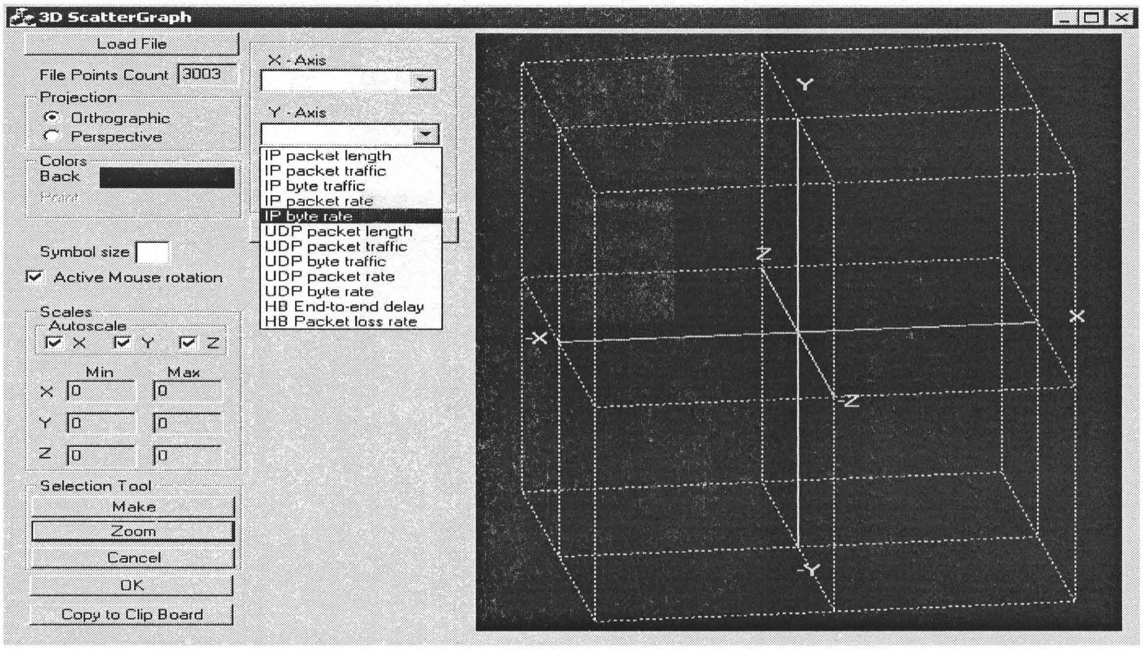
| | Min | Max |
|---|---|---|
| X | 0 | 1.0001 |
| Y | 0 | 1.0001 |
| Z | 0 | 1.0001 |

Look in: test

data
demo
test1
test11
test2
test3

File name:
Files of type:

## Figure 33: Assigning Network Parameters to Grid Axis



File Points Count 3003

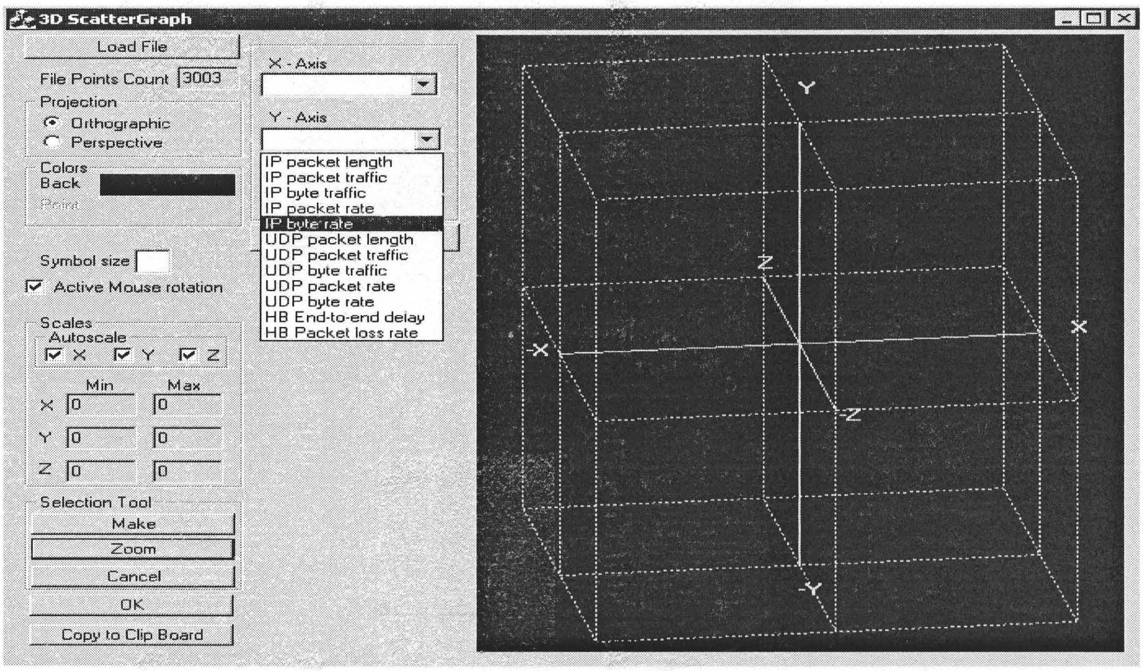
Projection: ● Orthographic ○ Perspective

☑ Active Mouse rotation

Autoscale: ☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

## Figure 34: Window setup for data



File Points Count 3003

Projection: ● Orthographic ○ Perspective

☑ Active Mouse rotation

Autoscale: ☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

## 3.3 Network Traffic Analysis with Scatter3D

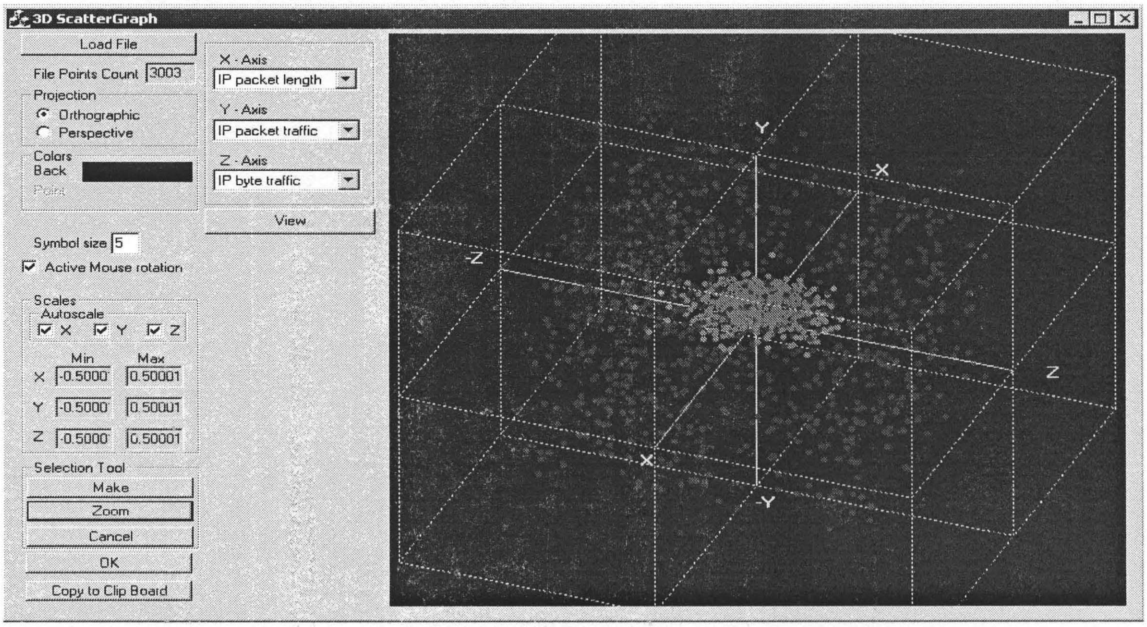
### 3.3.1 One Attack Data Files

The single attack stream or data file will have two different integers in the last column as a label. The lable is an integer that has a decimal value representing the color of the incident on the scatter plot. Below ia a sample data file showing the attack and the none attack records with a decimal coded label at the end of each record. The label is either blue for normal data records or pink for attack data records.

**Table 8: Single Attack Data for Scatter3D**

```
-0.049149 -0.043817 -0.593351 -0.848345  0.709313 -0.108074 -0.190108  0.460537 -0.922821 -0.436087  0.972837  0.761467 16711680
 0.317239  0.637969  0.137461  0.015612 -0.198789 -0.474188  0.199966  0.056807 -0.340171 -0.644028  0.000000  0.000000 16711935
 0.391402  0.379052  0.146744  0.218563  0.068243 -0.154856  0.136364  0.026927 -0.058556 -0.336626  0.000000  0.000000 16711680
 0.144728  0.177781  0.020388  0.063014 -0.020946 -0.069337  0.058745 -0.068485 -0.01168  -0.157757  0.000000  0.000000 16711680
-0.703927  0.080202 -0.137251 -0.061976 -0.535594  0.451147 -0.773159  0.411605  0.556766 -0.275018 -0.251367 -0.573110 16711680
 0.107513  0.104682 -0.943591  0.575237 -0.553928  0.171986 -0.651007 -0.507398 -0.876832  0.430235 -0.257824 -0.821946 16711680
 0.164872  0.192417  0.063283  0.029627 -0.01803  -0.003151  0.110798  0.042836 -0.058346 -0.111349  0.000000  0.000000 16711680
 0.747117  0.809016  0.601406  0.130965 -0.224100  0.018744  0.378346  0.182083 -0.328115 -0.693127  0.000000  0.000000 16711935
-0.944184  0.290334 -0.955873  0.350699 -0.819614  0.323562 -0.656014  0.618302  0.441047  0.270474 -0.555522 -0.200908 16711680
 0.187959  0.208043  0.114206  0.029943 -0.058109  0.021972  0.075604  0.003616 -0.09319  -0.174798  0.000000  0.000000 16711680
 0.434053  0.414970  0.280114  0.138378 -0.111335 -0.083319  0.345815  0.270825 -0.083166 -0.349351  0.000000  0.000000 16711680
 0.010215 -0.171472  0.749882 -0.836213  0.582469 -0.764543 -0.985447 -0.33555  -0.135984  0.345018  0.304784  0.511508 16711680
-0.236394 -0.835222  0.641857 -0.589062  0.029886  0.150342  0.273245  0.547866  0.781407  0.282123 -0.076981  0.232167 16711680
 0.334462  0.394018  0.228194  0.114896 -0.14569  -0.238689  0.076060  0.090446 -0.053336 -0.378295  0.000000  0.000000 16711680
 0.689185  0.770157  0.540454  0.313924 -0.250605  0.052820  0.552974  0.307618 -0.09374  -0.575341  0.000000  0.000000 16711935
 0.179705  0.173493  0.164853  0.000955 -0.050681 -0.044768  0.111615  0.090536 -0.081597 -0.194046  0.000000  0.000000 16711680
 0.697935 -0.129687  0.308189  0.896270 -0.174312 -0.505615 -0.566643 -0.952503 -0.254016  0.556431 -0.918911 -0.045830 16711680
 0.12536   0.940735 -0.586499  0.482003  0.389476 -0.214381 -0.600825 -0.269389  0.014457 -0.075249  0.991691 -0.023646 16711680
 0.182065  0.179006  0.078896  0.013751 -0.064946  0.007020  0.147368  0.114794 -0.104759 -0.197704  0.000000  0.000000 16711680
-0.927622 -0.492614 -0.435281 -0.018856 -0.302513  0.045763 -0.712160 -0.031581 -0.651917  0.859393  0.379144 -0.818468 16711680
 0.735239  0.738614  0.466809  0.035609 -0.143944 -0.057270  0.616653  0.550009 -0.09374  -0.49324   0.000000  0.000000 16711935
 0.167862  0.167629  0.065629  0.076648  0.010366 -0.048653  0.116374  0.046142  0.000003 -0.129088  0.000000  0.000000 16711680
-0.608057 -0.391288 -0.620058 -0.447768  0.12178  -0.925530 -0.329081  0.974692 -0.860079  0.903345  0.289158 -0.862329 16711680
 0.682182  0.690641  0.388895  0.174524 -0.111657  0.099757  0.755013  0.641853 -0.140615 -0.532956  0.000000  0.000000 16711935
 0.751563  0.790538  0.356056  0.086959 -0.292073 -0.114288  0.610613  0.355514 -0.140615 -0.721245  0.000000  0.000000 16711935
 0.56967   0.781351  0.285276  0.268981  0.012312 -0.354432  0.388265  0.220759 -0.040700 -0.583184  0.000000  0.000000 16711935
 0.590407  0.750068  0.198551  0.100932 -0.200989 -0.198893  0.559177  0.230249 -0.09374  -0.530989  0.000000  0.000000 16711935
 0.376057  0.496528  0.191715  0.183532 -0.158317 -0.160906  0.270261  0.177623 -0.105386 -0.438155  0.000000  0.000000 16711680
 0.081109  0.122444  0.009061 -0.021171 -0.039815 -0.121420  0.085855  0.011036 -0.010144 -0.133474  0.000000  0.000000 16711680
 0.179323  0.328548 -0.021295  0.112477  0.002998 -0.442612  0.177810 -0.124191  0.050454 -0.572747  0.000000  0.000000 16711935
 0.386531  0.075796 -0.155725  0.128416  0.078614 -0.569842 -0.050465 -0.225071  0.019204 -0.461186  0.000000  0.000000 16711935
 0.309781  0.210559  0.00595  -0.032536 -0.241038 -0.262320  0.034129 -0.057818 -0.105508 -0.454726  0.000000  0.000000 16711680
 0.082304  0.082576  0.000513 -0.027207 -0.067814 -0.103449  0.027335 -0.019961 -0.065219 -0.165289  0.000000  0.000000 16711680
-0.991067 -0.888219 -0.445267 -0.08541  -0.819128 -0.738939 -0.239291  0.357064 -0.001649  0.251235  0.412362  0.955205 16711680
 0.090919  0.085773 -0.035956  0.041853 -0.031493 -0.138274 -0.000692 -0.14762  -0.038033 -0.155536  0.000000  0.000000 16711680
 0.081649  0.093047 -0.035195  0.008614 -0.008551 -0.145783 -0.021524 -0.099471  0.009327 -0.132993  0.000000  0.000000 16711680
-0.60097   0.143413 -0.330393  0.524701 -0.928772 -0.803426 -0.468137  0.985284  0.244573 -0.968757  0.525605  0.260571 16711680
 0.420931  0.469677  0.212298  0.11715  -0.098598 -0.006823  0.381653  0.270204 -0.116326 -0.37335   0.000000  0.000000 16711680
 0.281995  0.244851 -0.006376  0.050594 -0.195977 -0.537557  0.064542 -0.195402 -0.043296 -0.431215  0.000000  0.000000 16711935
 0.310098  0.457406  0.000188  0.109845 -0.094451 -0.250760 -0.063073 -0.322889 -0.04171  -0.366315  0.000000  0.000000 16711680
 0.547840 -0.46884   0.310207 -0.683995  0.146900  0.733181 -0.988448 -0.908958  0.834848 -0.008057 -0.342280 -0.520760 16711680
 0.540163  0.617948  0.038513  0.026018 -0.154888 -0.652474 -0.027767 -0.052547 -0.262046 -0.654288  0.000000  0.000000 16711935
 0.427912  0.428198  0.234772  0.039455 -0.279615 -0.022898  0.353072  0.220889 -0.181573 -0.427578  0.000000  0.000000 16711680
 0.103212  0.14878   0.03891   0.035844 -0.040444 -0.075112  0.080813  0.039821 -0.006249 -0.133391  0.000000  0.000000 16711680
 0.562605  0.118813 -0.183079  0.031938 -0.685757  0.011515 -0.001012  0.60775   0.112086  0.018164  0.779682 -0.045400 16711680
 0.385408  0.466384  0.263139  0.069314 -0.10698  -0.043016  0.185828  0.172342 -0.111107 -0.425987  0.000000  0.000000 16711680
 0.396910  0.376252  0.105295  0.017306 -0.027869 -0.075836  0.35096   0.199342 -0.096869 -0.297959  0.000000  0.000000 16711680
 0.102334  0.119473  0.001952  0.023315  0.008827 -0.113807  0.047604 -0.032456 -0.033497 -0.148634  0.000000  0.000000 16711680
 0.098810  0.130975  0.038949  0.0052   -0.011108 -0.096139  0.049002 -0.012252 -0.038033 -0.1395    0.000000  0.000000 16711680
```

## Figure 35: Single Attack data scatter plot



3D ScatterGraph

File Points Count 3003

Projection
- ● Orthographic
- ○ Perspective

Colors
Back
Point

X - Axis: IP packet length

Y - Axis: IP packet traffic

Z - Axis: IP byte traffic

Symbol size 5

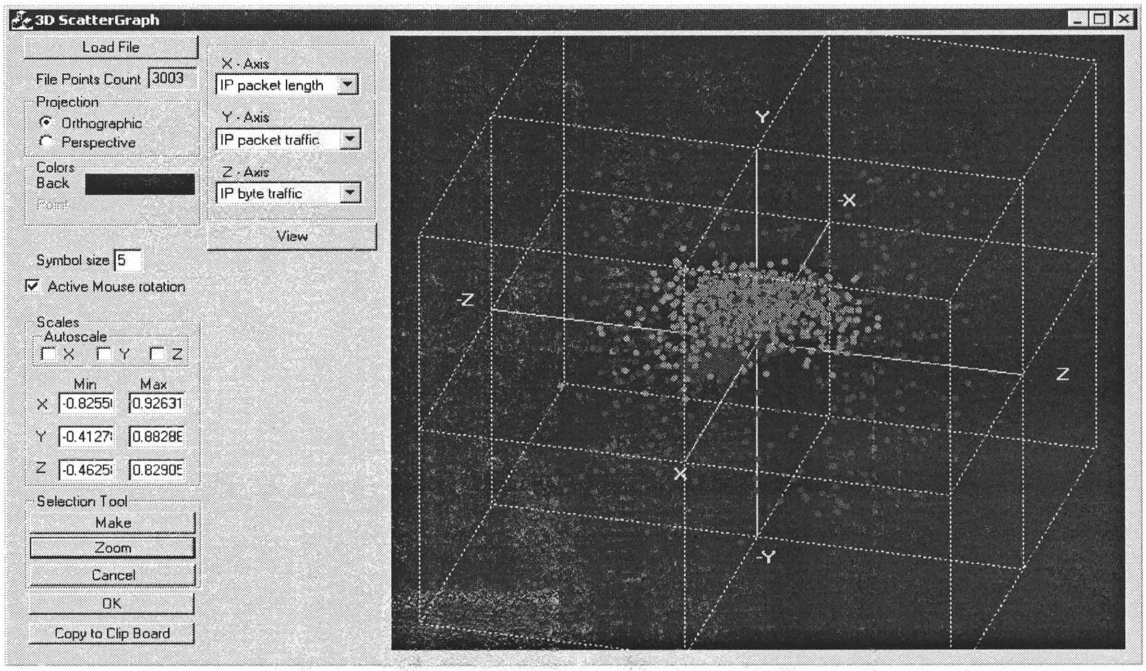
☑ Active Mouse rotation

Scales
Autoscale ☑ X ☑ Y ☑ Z

|  | Min | Max |
|---|---|---|
| X | -0.5000 | 0.50001 |
| Y | -0.5000 | 0.50001 |
| Z | -0.5000 | 0.50001 |

Selection Tool

## Figure 36: Closer look at the attack records



3D ScatterGraph

File Points Count 3003

Projection
- ● Orthographic
- ○ Perspective

Colors
Back
Point

X - Axis: IP packet length

Y - Axis: IP packet traffic

Z - Axis: IP byte traffic

Symbol size 5

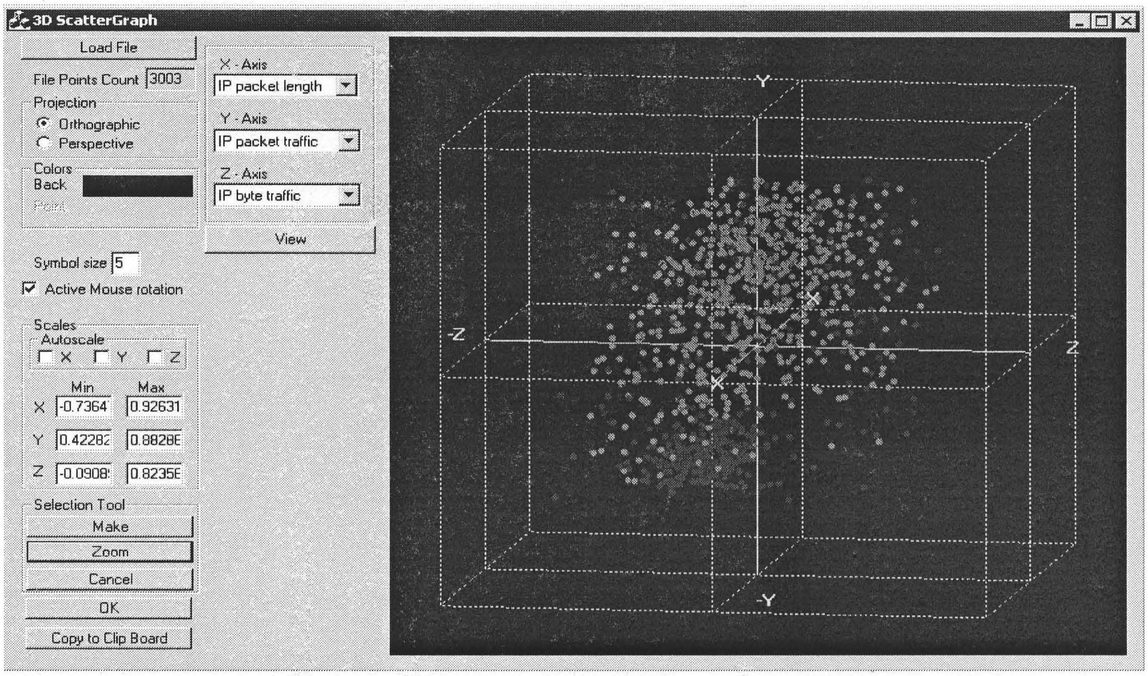
☑ Active Mouse rotation

Scales
Autoscale ☐ X ☐ Y ☐ Z

|  | Min | Max |
|---|---|---|
| X | -0.8255( | 0.92631 |
| Y | -0.4127( | 0.8828E |
| Z | -0.4625( | 0.82905 |

Selection Tool

Figure 37: Closer look at the attack records



File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

X - Axis: IP packet length
Y - Axis: IP packet traffic
Z - Axis: IP byte traffic

Symbol size 5

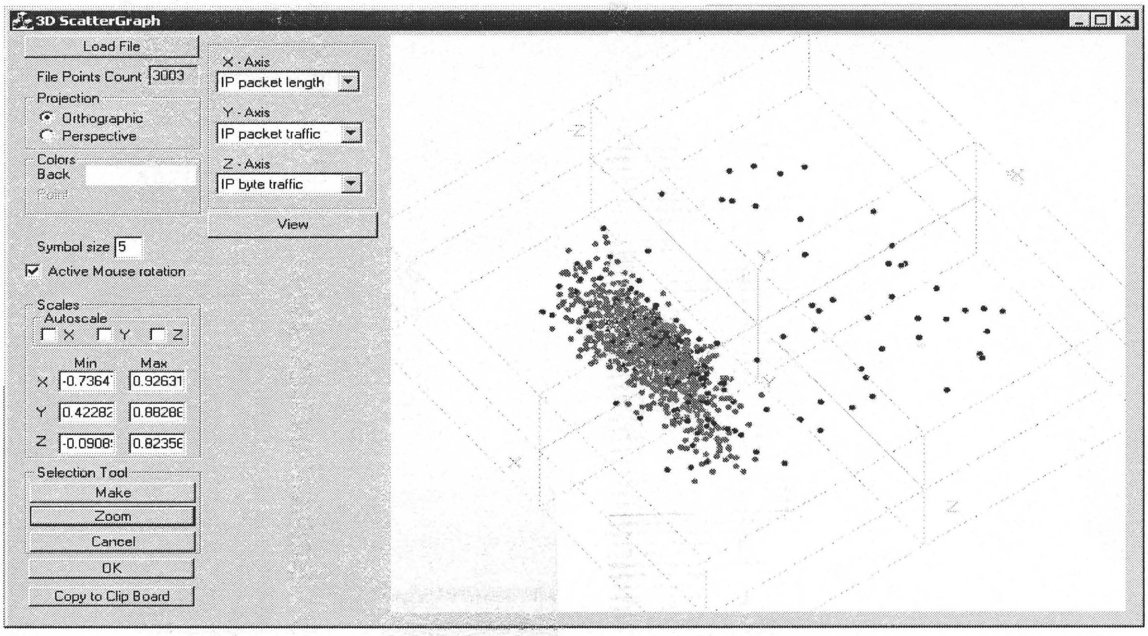
☑ Active Mouse rotation

Scales
Autoscale
□ X □ Y □ Z

| | Min | Max |
|---|---|---|
| X | -0.7364 | 0.92631 |
| Y | 0.42282 | 0.8828E |
| Z | -0.0908 | 0.8235E |

Selection Tool

Figure 38: Same Attack records with different background and different view



File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

X - Axis: IP packet length
Y - Axis: IP packet traffic
Z - Axis: IP byte traffic

Symbol size 5

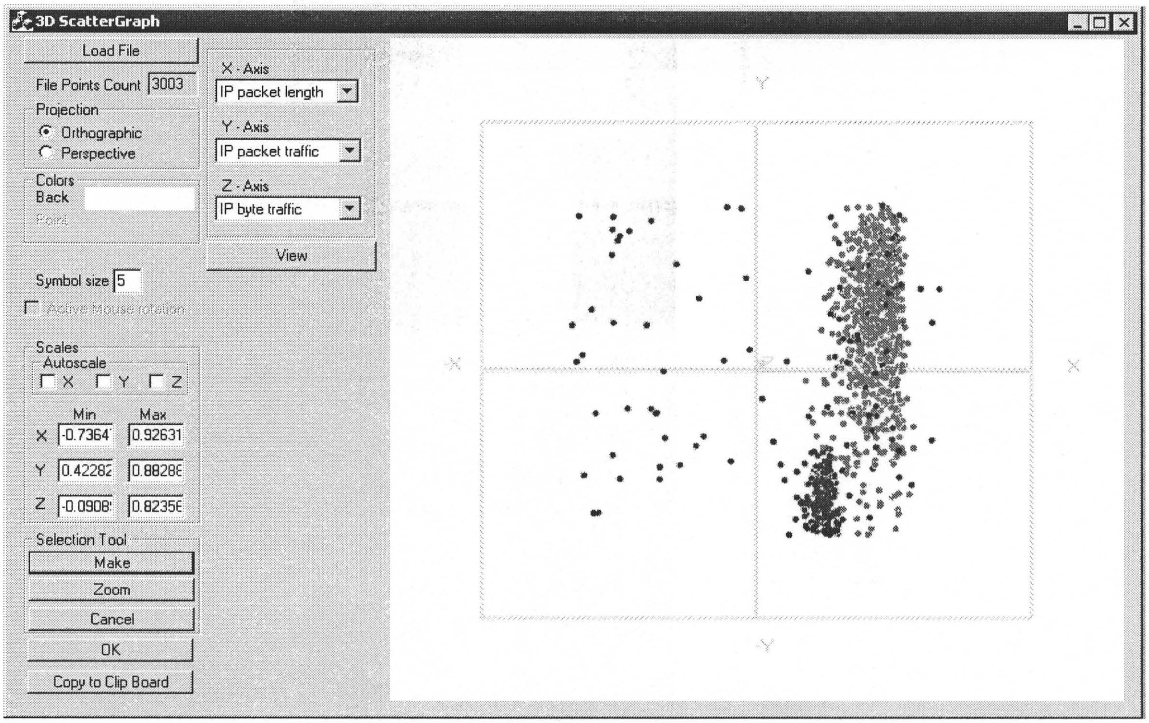
☑ Active Mouse rotation

Scales
Autoscale
□ X □ Y □ Z

| | Min | Max |
|---|---|---|
| X | -0.7364 | 0.92631 |
| Y | 0.42282 | 0.8828E |
| Z | -0.0908 | 0.8235E |

Selection Tool

## Figure 39: Parallax View in Scatter3D



File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

X - Axis: IP packet length

Y - Axis: IP packet traffic

Z - Axis: IP byte traffic

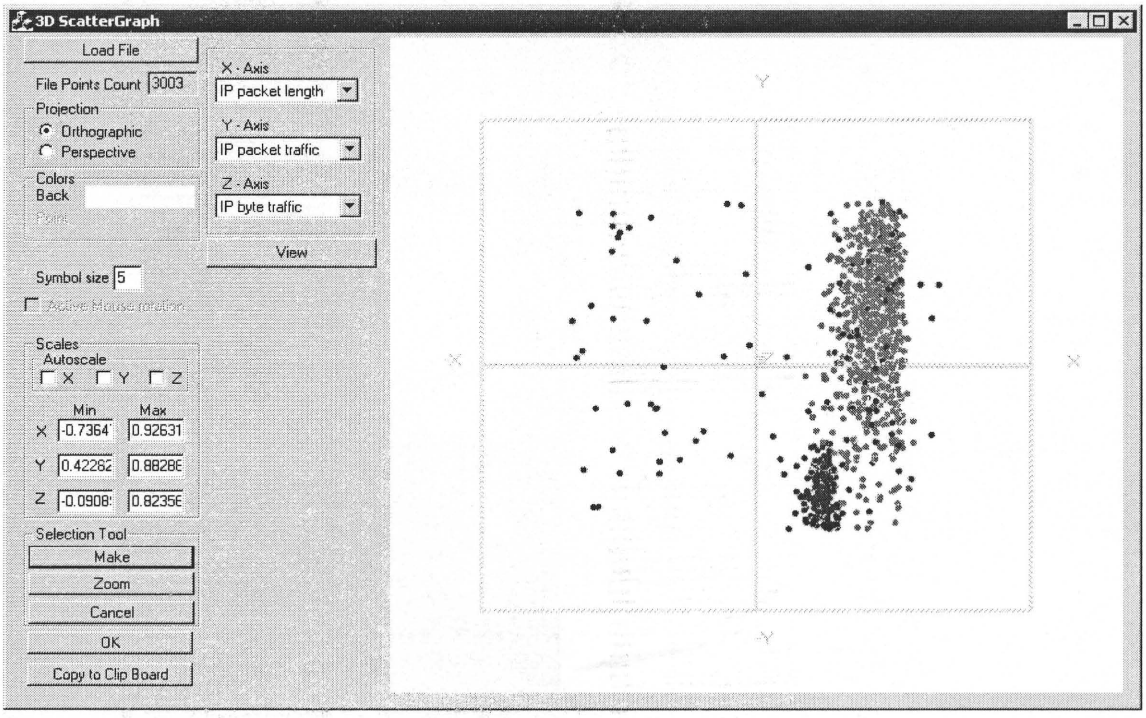
Symbol size 5

□ Active Mouse rotation

Scales
Autoscale
□ X □ Y □ Z

| | Min | Max |
|---|---|---|
| X | -0.7364 | 0.92631 |
| Y | 0.42282 | 0.88286 |
| Z | -0.0908 | 0.82356 |

## Figure 40: Two Dimensional View



File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

X - Axis: IP packet length

Y - Axis: IP packet traffic

Z - Axis: IP byte traffic

Symbol size 5

□ Active Mouse rotation

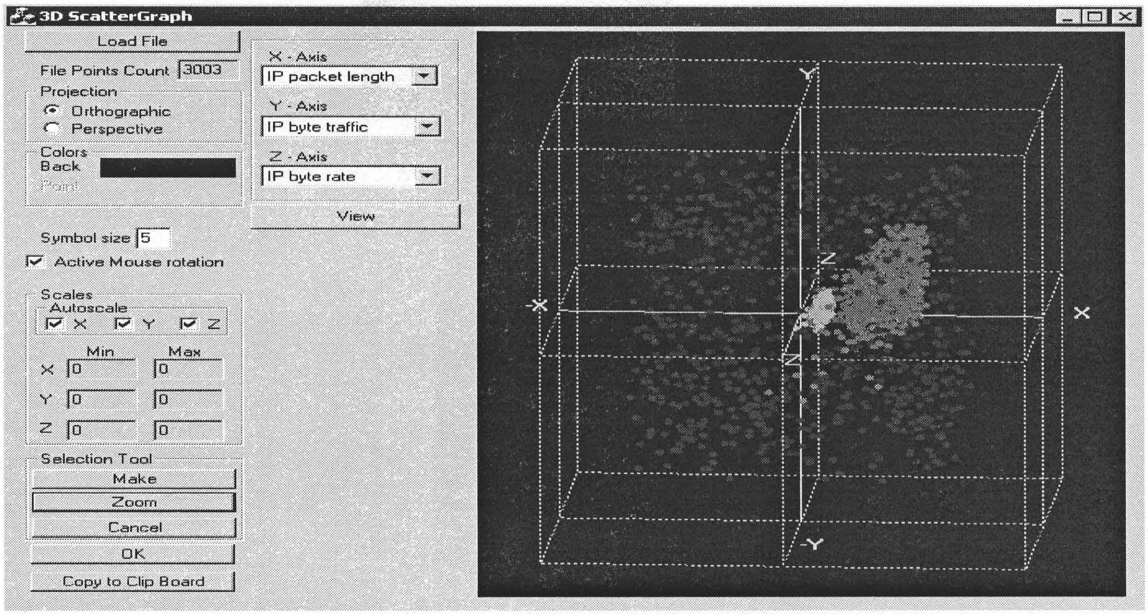
Scales
Autoscale
□ X □ Y □ Z

| | Min | Max |
|---|---|---|
| X | -0.7364 | 0.92631 |
| Y | 0.42282 | 0.88286 |
| Z | -0.0908 | 0.82356 |

### 3.3.2 Multiple Attack Data Files

In this section the data files carry more information that the previous one. The data label has more than two values. And could grow up to the maximum colors available.

**Figure 41: Multiple Attack Data Representation**



File Points Count 3003

Projection: ● Orthographic ○ Perspective

Symbol size 5

☑ Active Mouse rotation

Autoscale: ☑ X ☑ Y ☑ Z

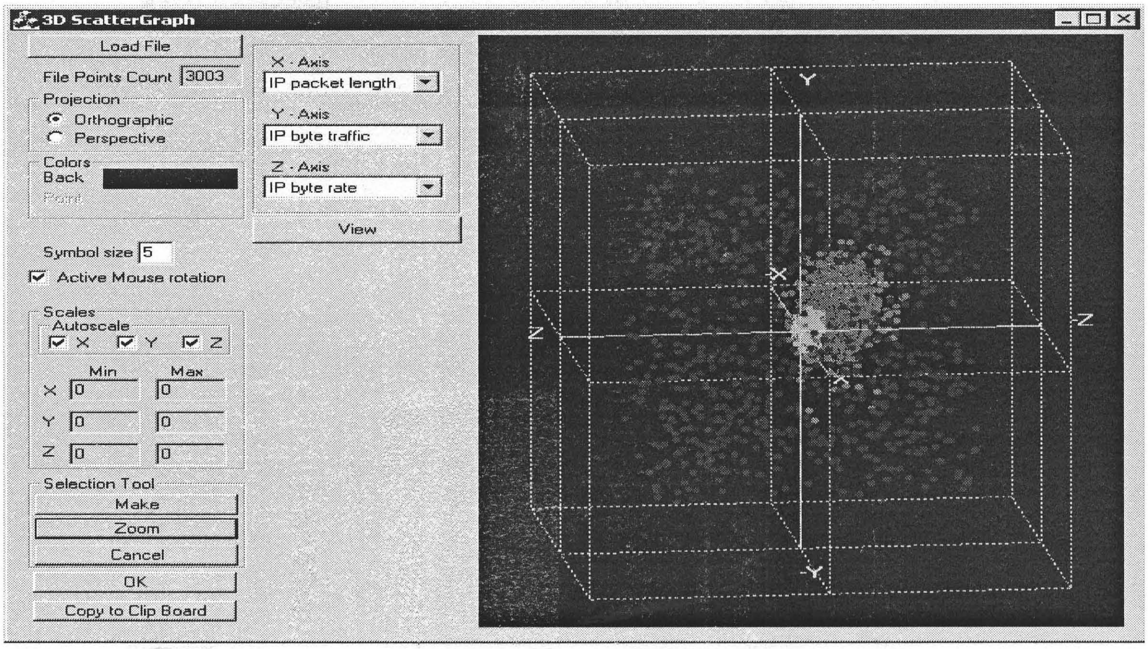
|   | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

X - Axis: IP packet length

Y - Axis: IP byte traffic

Z - Axis: IP byte rate

**Figure 42: Multiple Attack Data with Different view**



File Points Count 3003

Projection: ● Orthographic ○ Perspective

Symbol size 5

☑ Active Mouse rotation

Autoscale: ☑ X ☑ Y ☑ Z

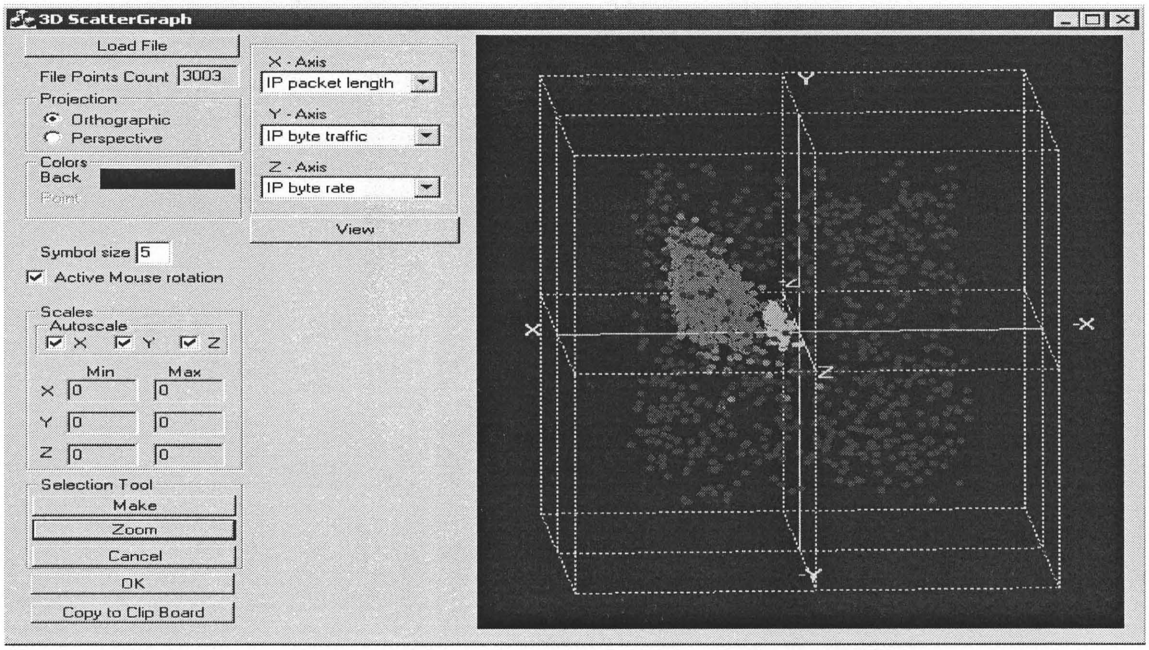
|   | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

X - Axis: IP packet length

Y - Axis: IP byte traffic

Z - Axis: IP byte rate

Figure 43: Rotation Showing the gaps between data types



3D ScatterGraph

File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

Symbol size 5

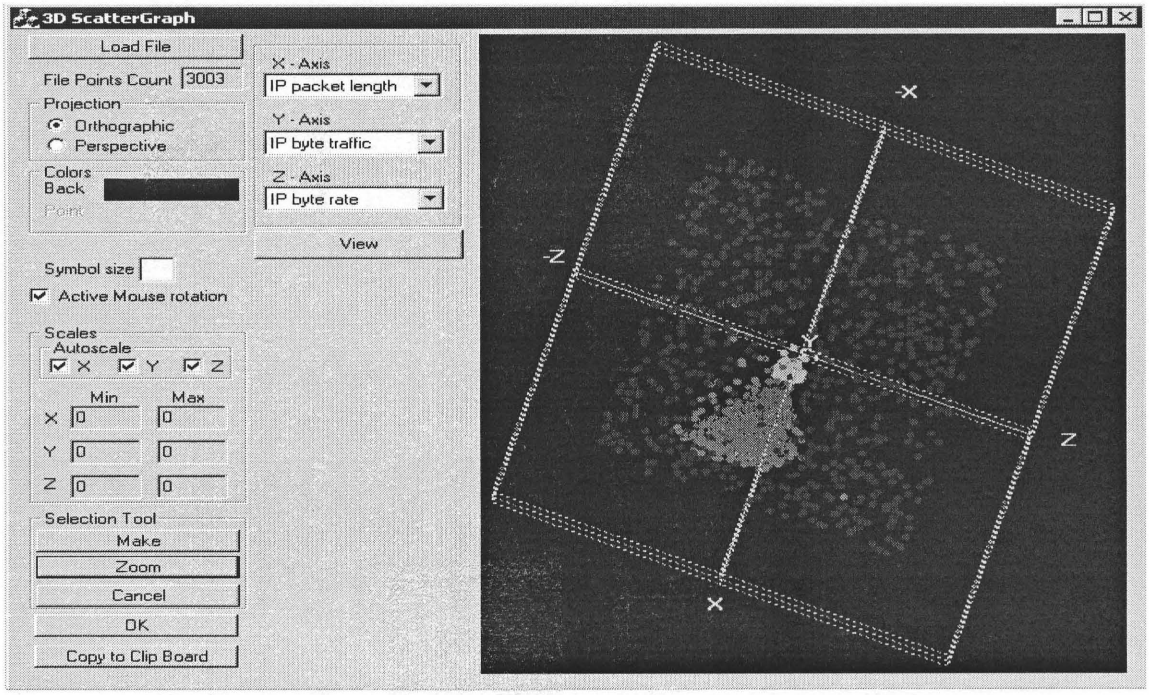
☑ Active Mouse rotation

Scales
Autoscale
☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

X - Axis: IP packet length
Y - Axis: IP byte traffic
Z - Axis: IP byte rate

Figure 44: Two Dimensional Representation of the same data



3D ScatterGraph

File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

Symbol size

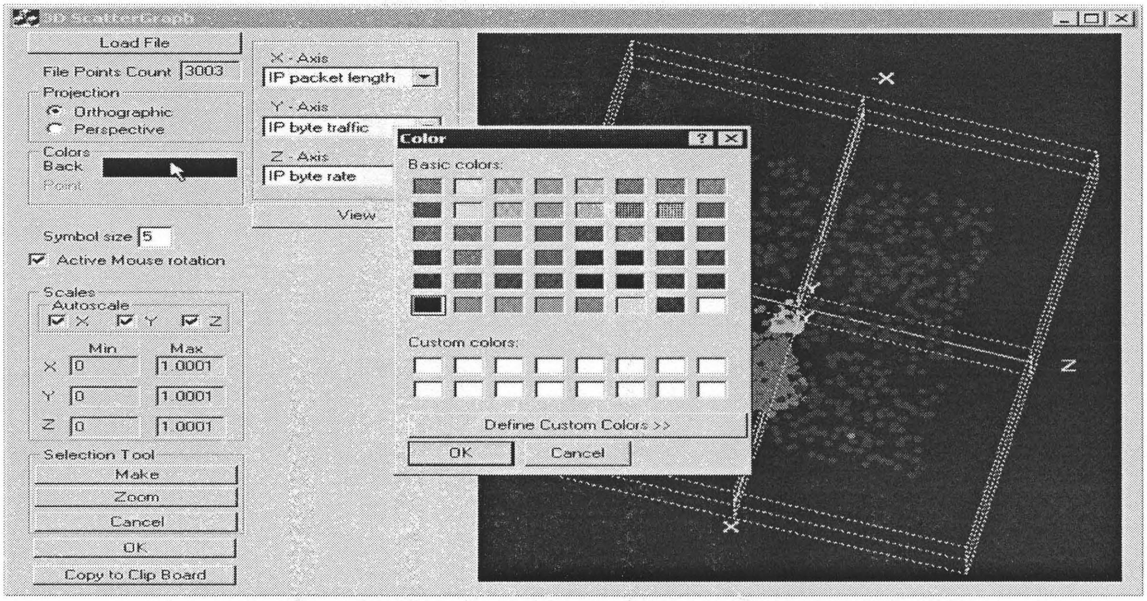
☑ Active Mouse rotation

Scales
Autoscale
☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 0 |
| Y | 0 | 0 |
| Z | 0 | 0 |

X - Axis: IP packet length
Y - Axis: IP byte traffic
Z - Axis: IP byte rate

## Figure 45: Changing the Data background color



3D ScatterGraph

File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

Symbol size 5

☑ Active Mouse rotation

Scales
Autoscale ☑ X ☑ Y ☑ Z

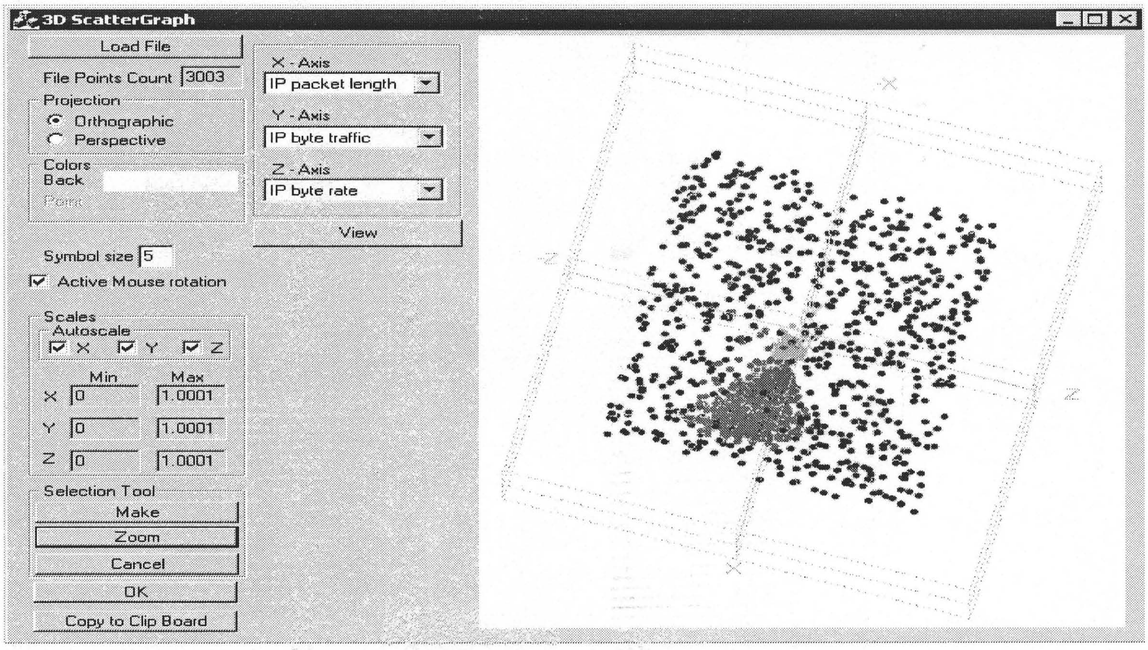
| | Min | Max |
|---|---|---|
| X | 0 | 1.0001 |
| Y | 0 | 1.0001 |
| Z | 0 | 1.0001 |

X - Axis: IP packet length

Y - Axis: IP byte traffic

Z - Axis: IP byte rate

## Figure 46: Same data with Different background



3D ScatterGraph

File Points Count 3003

Projection
● Orthographic
○ Perspective

Colors
Back
Point

Symbol size 5

☑ Active Mouse rotation

Scales
Autoscale ☑ X ☑ Y ☑ Z

| | Min | Max |
|---|---|---|
| X | 0 | 1.0001 |
| Y | 0 | 1.0001 |
| Z | 0 | 1.0001 |

X - Axis: IP packet length

Y - Axis: IP byte traffic

Z - Axis: IP byte rate

## 3.4 Further Work

The preliminary results from our experimental Scatter3D visualization give a positive indication of the potential offered by this tool, but a significant amount of research remains before it can function as a complete visualization system. A complete system will require the ability to directly receive inputs from a network data stream. The most difficult component of the analysis of network traffic by a neural network is the ability to effectively analyze the information in a live data stream. The various commands that are included in the data often provide the most critical element in the process of determining if an attack is occurring against a network. The most effective neural network architecture is also an issue that must be addressed. A feed- forward neural network that used a back propagation algorithm was chosen because of its simplicity and reliability in a variety of applications. However, alternatives such as the self- organizing feature map also possess advantages in attack detection that may promote their use. In addition, an effective neural network-based approach to attack detection must be highly adaptive. Most neural network architectures must be retrained if the system is to be capable of improving its analysis in response to changes in the input patterns, (e.g., "new" events are recognized with a consistent probability of being an attack until the network is retrained to improve the recognition of these events). Adaptive resonance theory ([2]) and self-organizing maps ([16]) offer an increased level of adaptability for neural networks, and these approaches are being investigated for possible use in an intrusion detection system. Finally, regardless of the initial implementation of a neural network-based intrusion detection system for attack detection it will be essential for the approach to be thoroughly tested in order to gain acceptance as a viable alternative to expert systems. Work has been conducted on taxonomies for testing intrusion detection systems ([3, 22]) that offer a standardized method of validating new technologies. Because of the questions that are certain to arise from the application of neural networks to intrusion detection, the use of these standardized methods is especially important.

# CHAPTER 4

# EXPERIMENTAL RESULTS AND DISCUSSIONS

There appear to be two primary reasons why neural networks have not been applied to the problem of attack detection in the past. The first reason relates to the training requirements of the neural network. Because the ability of the artificial neural network to identify indications of an intrusion is completely dependent on the accurate training of the system, the training data and the training methods that are used are critical. The training routine requires a very large amount of data to ensure that the results are statistically accurate. The training of a neural network for attack detection purposes may require thousands of individual attacks sequences, and this quantity of sensitive information is difficult to obtain. However, the most significant disadvantage of applying neural networks to intrusion detection is the "black box" nature of the neural network. Unlike expert systems, which have hard-coded rules for the analysis of events, neural networks adapt their analysis of data in response to the training, which is conducted on the network. The connection weights and transfer functions of the various network nodes are usually frozen after the network has achieved an acceptable level of success in the identification of events. While the network analysis is achieving a sufficient probability of success, the basis for this level of accuracy is not often known. The "Black Box Problem" has plagued neural networks in a number of applications [11]. This is an on-going area of neural network research.

There are two general implementations of neural networks in misuse detection systems. The first involves incorporating them into existing or modified expert systems. Unlike the previous attempts to use neural networks in anomaly detection by using them as replacements for existing statistical analysis components, this proposal involves using the neural network to filter the incoming data for suspicious events which may be indicative of misuse and forward these events to the expert system. This configuration should improve the effectiveness of the detection system by reducing the false alarm rate of the expert system. Because the neural network will determine a probability that a particular event is indicative of an attack, a threshold can be established where the event is forwarded to the expert system for additional analysis. Since the expert system is only receiving data on events, which are viewed as suspicious, the sensitivity of the expert system can be increased, (typically, the sensitivity of expert systems must be kept low to reduce the incidence of false alarms). This configuration would be beneficial to organizations that have invested in rule-based expert system technology by improving the effectiveness of the system while it preserves the investment that has been made in existing intrusion detection systems. The disadvantage of this approach would be that as the neural network improved its ability to identify new attacks the expert system would have to be updated to also recognize these as threats. If the expert system were not updated then the new attacks identified by the neural network would increasingly be ignored by the expert system because its rule-base would not be capable of recognizing the new threat. The second approach would involve the neural network as a standalone misuse detection system. In this configuration, the neural network would receive data from the network stream and analyze the information for instances of misuse. Any instances, which are identified as indicative of attack would be forwarded to a security administrator or used by an automated intrusion response system. This approach would offer the benefit of speed over the previous approach, since there would only be a single layer of analysis. In addition, this configuration should improve in effectiveness over time as the network learns the characteristics of attacks. Unlike the first approach, this concept would not be limited by the analytical ability of the expert system, and as a result, it would be able to expand beyond the limits of the expert system's rule-base.

In an effort to determine the applicability of neural networks to the problem of misuse detection we conducted an analysis the approach utilizing simulated network traffic. The experiment was designed to determine if indications of attack could be identified from typical network traffic, but it was not intended to completely resolve the issue of applying neural networks to misuse detection. The analysis did not address the potential benefit of identifying a priority attacks that may be possible through the use of neural networks. However, determining if a neural network was capable of identifying misuse incidents with a reasonable degree of accuracy was considered to be the first step in applying the technology to this form of intrusion detection.

The first prototype neural network was designed to determine if a neural network was capable of identifying specific events that are indications of misuse. Neural networks had been shown to be capable of identifying TCP/IP network events in [27], but our prototype was designed to test the distinction between various network traffic which in return will lead to facilitate the use of neural network to identify indications of attack. The prototype utilized a server client architecture that consisted of fully connected layers with nine input nodes. Our prototype was designed to collect various traffic parameters described in the previous chapters. The prototype was configured to capture the data for each event, which would be consistent with a network frame, (e.g., source address, destination address, packet data, etc.). In addition to the "normal" network activity that was collected as events, the host for the monitor was "attacked" using the coded programs residing at the clients (nodes). These applications (Opnet) were used because of their ability to generate a large number of simulated attacks against a specified network host. Opnet were configured for a variety of attacks, ranging from denial of service attacks to port scan. Approximately 12000 individual PDF's were collected and stored in data files of which approximately in some cases 10000 were simulated attacks.

Three levels of preprocessing of the data were conducted to prepare the data for visualization and then for use in the training and testing of the neural network. In the first round of preprocessing random data records were selected from the available pdf's files. The selected records are typically present in network data packets and they provide a complete description of the information transmitted by the packets. Since the data records are long a compression scheme was needed to convert the records to standard numeric presentation. The second part of the preprocessing phrase consisted of converting data elements (Attack type, and Raw data) into a standardized numeric representation. The process involved the creation of matlab wavelet code.

The program computes the similarity values between two PDF's. It is assumed that the input is of the form: 8 real numbers, with numbers 5, 6, and 7 denoting either an attack or typical behavior (with the sum being equal to zero for typical, non-zero for attack), then followed by 12 sets of 67, with each set being a PDF. The 67 numbers are of the form: first is the sample size, second is the minimum value, third is the bin width (for uniform bins), followed by 64 percentile values corresponding to the probability of each bin. The similarity measures are computed for five different levels of wavelet computations. The output consists of 12 similarity measures and one flag indicating either typical behavior or attack traffic. The output is then stored in data files for the visualization and neural network use. The data files were used during training and testing of the neural network.

The training/testing iterations of the neural network required some time to complete. At the conclusion of the training the following results were obtained: The figures matched very closely with the desired. After the completion of the training and testing of the traffic various connection weights were frozen and the network was interrogated. Three sample patterns containing "normal" network events and a single simulated attack event were used to test the neural network. While this prototype (visualization) was not designed to be a complete intrusion detection system, the results clearly demonstrate the potential of a the tools used and neural network to detect individual instances of possible attack from a representative network data stream.

# CHAPTER 5

# CONCLUSIONS

Research and development of intrusion detection systems has been ongoing since the early 1980's and the challenges faced by designers increase as the targeted systems because more diverse and complex. Misuse detection is a particularly difficult problem because of the extensive number of vulnerabilities in computer systems and the creativity of the attackers. Neural networks provide a number of advantages in the detection of these attacks. The early results of our tests of these technologies show significant promise, and our future work will involve the refinement of this approach and the development of a full-scale demonstration system.

The framework presented here is for the evaluation and visualizing techniques for network traffic characterization. I have applied my methodology to twelve target metrics: IP packet length, IP in packet traffic, IP in byte traffic, IP in packet rate, IP in byte rate, UDP in packet length, UDP in packet traffic, UDP in byte traffic, UDP in packet rate, UDP in byte rate, Heart Beat End to end delay, and Heart Beat Packet loss rate. My experimental data consisted of a packet trace obtained from Opnet simulation. Because the characteristics of our population of network data collected some experimental parameters were controlled. Neural network sigmoid where used to evaluate the goodness of the data and to produce more similar data. One important result is that the data collected using the neural network sigmoid and the simulated data are compatible with the original distribution of the real network data with consideration to the inter-arrival time and the packet size.

My methodology can be extended to and applied to characterization of network traffic that is based on proportions, e.g., TCP/UDP port distribution. More difficult would be to characterize the goodness of fit of the sampled source destination traffic matrix, mainly because of its large size and because many traffic pairs generate small amounts of traffic typical sampling intervals.

# APPENDIX SCATTER3D CODE

## GLMouseRotate.cpp

```
#include "stdafx.h"
#include "GLMouseRotate.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/*
This class is a simple extension that allows the mouse to rotate the
image.
*/
/////////////////////////////////////////////////////////////////////////////
// CGLMouseRotate

CGLMouseRotate::CGLMouseRotate()
{
	m_bAllowMouseRotate=TRUE;
	m_bInMouseRotate=FALSE;

	m_xMouseRotation = 0.0f;
	m_yMouseRotation = 0.0f;

}

CGLMouseRotate::~CGLMouseRotate()
{
}


BEGIN_MESSAGE_MAP(CGLMouseRotate, COpenGLWnd)
	//{{AFX_MSG_MAP(CGLMouseRotate)
	ON_WM_LBUTTONDOWN()
	ON_WM_LBUTTONUP()
	ON_WM_MOUSEMOVE()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGLMouseRotate message handlers
```

```
void CGLMouseRotate::OnLButtonDown(UINT nFlags, CPoint point)
{
        if (m_bAllowMouseRotate)
        {
                m_LeftDownPos = point;
                SetCapture();
                ::SetCursor(LoadCursor(NULL, IDC_SIZEALL));
// NOTE: we need a flag, can't just check whether got capture,
// since capture might be set in a derived class for another reason
                m_bInMouseRotate=TRUE;
        }
        COpenGLWnd::OnLButtonDown(nFlags, point);
}

void CGLMouseRotate::OnLButtonUp(UINT nFlags, CPoint point)
{
        m_LeftDownPos = CPoint(0,0);        // forget where we clicked
        SetMouseCursor(     AfxGetApp()->LoadStandardCursor(IDC_ARROW));
        ReleaseCapture();
        m_bInMouseRotate=FALSE;

        COpenGLWnd::OnLButtonUp(nFlags, point);
}
void CGLMouseRotate::OnMouseMove(UINT nFlags, CPoint point)
{
// check if we are in mouse rotate
// can't just check if got capture, because might be captured in derived class for another
reason
        if (m_bInMouseRotate)
        {
                ASSERT(GetCapture()==this);
                ::SetCursor(LoadCursor(NULL, IDC_SIZEALL));
                m_yMouseRotation -= (float)(m_LeftDownPos.x - point.x)/3.0f;
                m_xMouseRotation -= (float)(m_LeftDownPos.y - point.y)/3.0f;
                m_LeftDownPos = point;
                InvalidateRect(NULL,FALSE);
        }

        COpenGLWnd::OnMouseMove(nFlags, point);
}

void CGLMouseRotate::DoMouseRotate()
{
        glRotatef(m_xMouseRotation, 1.0, 0.0, 0.0);
        glRotatef(m_yMouseRotation, 0.0, 1.0, 0.0);
}
```

## GLScatterGraph.cpp

```
#include "stdafx.h"
#include "GLScatterGraph.h"

#include <float.h>
#include <math.h>


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif



// returns the next biggest number from val set to sig significant figures
// use for top label axes
// thus 250=(247,2); 300 = (247,1)
float NextAbove(float val,int sig)
{
        float x=int(log10(fabs(val))+1)-sig;
        float mult=pow(10,x);
        return ceil(val/mult)*mult;
  // floor
}

float NextBelow(float val,int sig)
{
        float x=int(log10(fabs(val))+1)-sig;
        float mult=pow(10,x);
        return floor(val/mult)*mult;
}

/////////////////////////////////////////////////////////////////////////////
// CGLScatterGraph

CGLScatterGraph::CGLScatterGraph()
{
        m_xMouseRotation=10.0;
        m_yMouseRotation=-15.0;
        m_ProjType=0;
        m_SymbolSize=3;

        m_MaxX=1;
        m_MaxY=1;
```

```
    m_MaxZ=1;
    m_MinX=0;
    m_MinY=0;
    m_MinZ=0;
    m_bAutoScaleX=TRUE;
    m_bAutoScaleY=TRUE;
    m_bAutoScaleZ=TRUE;

    m_Colour=RGB(255,0,0);
}

CGLScatterGraph::~CGLScatterGraph()
{
}

void CGLScatterGraph::RenderData()
{
    int i;
    float xW=m_MaxX-m_MinX;
    float yW=m_MaxY-m_MinY;
    float zW=m_MaxZ-m_MinZ;
// need to scale data between -0.5 and 0.5 in all dimensions
    float x,y,z,r,g,b;

// NOTE: z axis goes -ve into screen, so invert z values to make
// like normal 3D graph (z gets bigger going into screen
    if (m_pColList==NULL)       // all the same colour
    {
        r=float(GetRValue(m_Colour))/255;
        g=float(GetGValue(m_Colour))/255;
        b=float(GetBValue(m_Colour))/255;
        glColor3f(r,g,b);
        glBegin(GL_POINTS);
        for (i=0; i<m_Count; i++)
        {
            if (PtWithinAxes(m_pDat[i*3],m_pDat[i*3+1],m_pDat[i*3+2]))
            {
                x=(m_pDat[i*3]-m_MinX)/xW-0.5;
                y=(m_pDat[i*3+1]-m_MinY)/yW-0.5;
                z=(m_pDat[i*3+2]-m_MinZ)/zW-0.5;
                glVertex3f(x,y,-z);
//  TRACE("real world coords in RenderData = %f, %f, %f\n",x,y,z);
            }
        }
        glEnd();
    }
    else // got separate colours for each point
    {
        COLORREF currentCol;
        i=0;
        currentCol=m_pColList[i];
        while (i<m_Count)
        {
            r=float(GetRValue(currentCol))/255;
            g=float(GetGValue(currentCol))/255;
            b=float(GetBValue(currentCol))/255;
            glColor3f(r,g,b);
            glBegin(GL_POINTS);
            while (i<m_Count)
            {
                if (m_pColList[i]!=currentCol)
                {
                    currentCol=m_pColList[i];
                    glEnd();
                    break;
                }
                if
(PtWithinAxes(m_pDat[i*3],m_pDat[i*3+1],m_pDat[i*3+2]))
                {
                    x=(m_pDat[i*3]-m_MinX)/xW-0.5;
                    y=(m_pDat[i*3+1]-m_MinY)/yW-0.5;
                    z=(m_pDat[i*3+2]-m_MinZ)/zW-0.5;
                    glVertex3f(x,y,-z);
                }
                i++;
//TRACE("real world coords in RenderData = %f, %f, %f\n",x,y,z);
            }
        }
        glEnd();
    }
    Invalidate();
}


BEGIN_MESSAGE_MAP(CGLScatterGraph, CGLMouseRotate)
    //{{AFX_MSG_MAP(CGLScatterGraph)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
```

```
// CGLScatterGraph message handlers
void CGLScatterGraph::OnDrawGL()
{
#ifdef _DEBUG
        int rv;
        glGetIntegerv(GL_RENDER_MODE,&rv);
        CString mode;
        if (rv==GL_RENDER)
                mode="Render";
        else if (rv==GL_FEEDBACK)
                mode="Feedback";
        else
                mode="Unknown";
TRACE("In CGLScatterGraph::OnDrawGL, mode = %s\n",mode);
#endif

        glPushMatrix();         // this saves the modelmatrix settings

        DoMouseRotate();
        m_GraphBox.Draw(); // draw box from display list constructed in OnCreate
        glPointSize(m_SymbolSize);
        RenderData();

        glPopMatrix();// this restores the modelmatrix settings to before translation etc
}

void CGLScatterGraph::OnDrawGDI(CPaintDC *pDC)
{
TRACE("In CGLScatterGraph::OnDrawGDI\n");
        CRect r;
        GetClientRect(&r);
        CPen pen(PS_SOLID,0,RGB(255,255,255));
        CPen *oP=pDC->SelectObject(&pen);
        pDC->MoveTo(r.TopLeft());
        pDC->LineTo(r.BottomRight());
        pDC->SelectObject(oP);
        ;
}
void CGLScatterGraph::OnSizeGL(int cx, int cy)
{
// set correspondence between window and OGL viewport
                glViewport(0,0,cx,cy);

                SetProjection(m_ProjType);
#if 0
// update the camera
```

```
        glPushMatrix();
            glMatrixMode(GL_PROJECTION);
                glLoadIdentity();
                if (m_ProjType==0)  // ortho
                    glOrtho(-1,1,-1,1,2.0,10.);     // need BIGGER
Frustum for ortho
                else
                    gluPerspective(25,GetAspectRatio(),1,15.0);
//      glFrustum(-0.5,0.5,-0.5,0.5,2.0,10.);
                glTranslatef(0.0f,0.0f,-4.f);
            glMatrixMode(GL_MODELVIEW);
        glPopMatrix();
#endif
}

void CGLScatterGraph::OnCreateGL()
{
// following could also be set by calling COpenGLWnd::OnCreateGL();
// perform hidden line/surface removal (enabling Z-Buffer)
    glEnable(GL_DEPTH_TEST);
// set background color to black
    glClearColor(0.f,0.f,0.f,1.0f );
// set clear Z-Buffer value
    glClearDepth(1.0f);

// specific to scattergraph
    MakeFont();
//  RasterFont();  // call to instantiate font bitmaps (maybe should be in base class?)
// make a graph box, since we will need one for all time
    m_GraphBox.StartDef();// <- do not execute list immediately
        glColor3f(0.7f, 0.7f, 0.7f);
        CPoint3D pt;
        //CPoint3D orgPt(-0.5,-0.5,0.5);
  CPoint3D orgPt(-0,-0,0); // point of the origin
        glBegin(GL_LINES);
////////////////////////////////
// main X
        pt=orgPt;
  //pt.Translate(-1.0,0,0);
        pt.Translate(-0.75,0,0);//0.1
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(1.5,0,0);//1.2
        glVertex3f(pt.x,pt.y,pt.z);
////////////////////////////////
// main Y
        pt=orgPt;
```

```
                pt.Translate(0,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,1.5,0);
                glVertex3f(pt.x,pt.y,pt.z);
/////////////////////////////////
// main Z
                pt=orgPt;
                pt.Translate(0,0,0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,-1.5);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();
/////////////////////////////////
// first Quadrant +X, +Y, +Z. first Cube in  3D
                glLineStipple(2,0xAAAA);
                glEnable(GL_LINE_STIPPLE);
   ///////////////////////////
                glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,-0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();
      ///////////////////////////
                glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0.75,0,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,-0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(-0.75,0,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();
      ///////////////////////////
                glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0.75,0,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();
```

```
glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(0.75,0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,-0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,0.75,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,0.75,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,0.75,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,0.75,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,-0.75,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,-0.75,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,-0.75,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINES);
        pt=orgPt;
        pt.Translate(0,-0.75,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(-0.75,0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,-0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(0.75,0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,-0.75,0);
```

```
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();

glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0.75,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();

glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(-0.75,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();

glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(-0.75,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,-0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();

glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0.75,-0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0,-0.75);
                glVertex3f(pt.x,pt.y,pt.z);
                pt.Translate(0,0.75,0);
                glVertex3f(pt.x,pt.y,pt.z);
                glEnd();

glBegin(GL_LINE_STRIP);
                pt=orgPt;
                pt.Translate(0.75,0,0);
```

```
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(0,0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(-0.75,0,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,-0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();

glBegin(GL_LINE_STRIP);
        pt=orgPt;
        pt.Translate(-0.75,0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,0,-0.75);
        glVertex3f(pt.x,pt.y,pt.z);
        pt.Translate(0,-0.75,0);
        glVertex3f(pt.x,pt.y,pt.z);
        glEnd();
```

```
////////////////////////////////
/////////////// Cube -X, -Y, +Z.
      glBegin(GL_LINE_STRIP);
                  pt=orgPt;
                  pt.Translate(0,-0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(-0.75,0,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  glEnd();

      glBegin(GL_LINE_STRIP);
                  pt=orgPt;
                  pt.Translate(0,0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0,0.75);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,-0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  glEnd();
      glBegin(GL_LINE_STRIP);
                  pt=orgPt;
                  pt.Translate(0,-0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0,0.75);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  glEnd();

      glBegin(GL_LINE_STRIP);
                  pt=orgPt;
                  pt.Translate(0,-0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0,-0.75);
                  glVertex3f(pt.x,pt.y,pt.z);
                  pt.Translate(0,0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
                  glEnd();

      glBegin(GL_LINE_STRIP);
                  pt=orgPt;
                  pt.Translate(0,-0.75,0);
                  glVertex3f(pt.x,pt.y,pt.z);
```

```
                    pt.Translate(0.75,0,0);
                    glVertex3f(pt.x,pt.y,pt.z);
                    pt.Translate(0,0.75,0);
                    glVertex3f(pt.x,pt.y,pt.z);
                    glEnd();

//////////////////////////////
// Labeling the Axis
//
 glEnd();
 glDisable(GL_LINE_STIPPLE);

                    pt=orgPt;
                    float endAx = 0.85;
                    float startAx = -0.85;

                    pt.Translate(startAx,0,0);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("-X");
                    pt=orgPt;
                    pt.Translate(0,startAx,0);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("-Y");
                    pt=orgPt;
                    pt.Translate(0,0,-startAx);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("-Z");
      pt=orgPt;
                    pt.Translate(endAx,0,0);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("X");
                    pt=orgPt;
                    pt.Translate(0,endAx,0);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("Y");
                    pt=orgPt;
                    pt.Translate(0,0,-endAx);
                    glRasterPos3f(pt.x,pt.y,pt.z);
                    PrintString("Z");
             m_GraphBox.EndDef();

        glEnable(GL_POINT_SMOOTH);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
//                      glHint(GL_POINT_SMOOTH_HINT,GL_NICEST);
}
```

```
void CGLScatterGraph::SetProjection(int type)
{
        m_ProjType=type;
        BeginGLCommands();
        glPushMatrix();
                glMatrixMode(GL_PROJECTION);
                        glLoadIdentity();
                        if (m_ProjType==0)   // ortho
                                glOrtho(-1,1,-1,1,2.0,10.);      // need BIGGER Frustum for
ortho
                        else
                                gluPerspective(25,GetAspectRatio(),1,15.0);
//              glFrustum(-0.5,0.5,-0.5,0.5,2.0,10.);
                        glTranslatef(0.0f,0.0f,-4.f);
                glMatrixMode(GL_MODELVIEW);
        glPopMatrix();
        EndGLCommands();
}

void CGLScatterGraph::SetData(int count,COLORREF col,float *pCoords, COLORREF
*pColList)
{
        m_Count=count;
        m_Colour=col;
        m_pDat=pCoords;
        m_pColList=pColList;
        AutoScale();
}

void CGLScatterGraph::AutoScale(int *pDoList)
{
        int i;
// find max & min
        float xMax=-FLT_MAX;
        float xMin=FLT_MAX;
        float yMax=-FLT_MAX;
        float yMin=FLT_MAX;
        float zMax=-FLT_MAX;
        float zMin=FLT_MAX;

        if (m_bAutoScaleX || m_bAutoScaleY || m_bAutoScaleZ)
        {
                for (i=0; i<m_Count; i++)
                {
                        if (pDoList!=NULL && pDoList[i]==0)
                continue;
            xMax=max(m_pDat[i*3],xMax);
            xMin=min(m_pDat[i*3],xMin);
            yMax=max(m_pDat[i*3+1],yMax);
            yMin=min(m_pDat[i*3+1],yMin);
            zMax=max(m_pDat[i*3+2],zMax);
            zMin=min(m_pDat[i*3+2],zMin);
        }
        if (m_bAutoScaleX)
        {
            m_MaxX=NextAbove(xMax,5);
            m_MinX=NextBelow(xMin,5);
        }
        if (m_bAutoScaleY)
        {
            m_MaxY=NextAbove(yMax,5);
            m_MinY=NextBelow(yMin,5);
        }
        if (m_bAutoScaleZ)
        {
            m_MaxZ=NextAbove(zMax,5);
            m_MinZ=NextBelow(zMin,5);
        }
    }
    Invalidate();
}

BOOL CGLScatterGraph::PtWithinAxes(float x,float y,float z)
{
    if ( x >= m_MinX &&
  x <= m_MaxX &&
  y >= m_MinY &&
  y <= m_MaxY &&
  z >= m_MinZ &&
  z <= m_MaxZ
 )
        return TRUE;
    return FALSE;
}
```

## GLSelectableScatterGraph.cpp

```
#include "stdafx.h"
#include "GLSelectableScatterGraph.h"
#include <float.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


CGLSelectableScatterGraph::CGLSelectableScatterGraph()
{
	m_bMakeSel=FALSE;
	m_SelPts.SetSize(0,100);

//	test_num=0;
}

CGLSelectableScatterGraph::~CGLSelectableScatterGraph()
{
	m_SelPts.RemoveAll();
}


BEGIN_MESSAGE_MAP(CGLSelectableScatterGraph, CGLScatterGraph)
	//{{AFX_MSG_MAP(CGLSelectableScatterGraph)
	ON_WM_LBUTTONDOWN()
	ON_WM_LBUTTONUP()
	ON_WM_MOUSEMOVE()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CGLSelectableScatterGraph message handlers
void CGLSelectableScatterGraph::StartMakeSel()
{
	m_bMakeSel=TRUE;
	m_bOldAllowRotate=m_bAllowMouseRotate;
	m_bAllowMouseRotate=FALSE;
}
void CGLSelectableScatterGraph::CancelSel()
{
	m_bMakeSel=FALSE;
```

```
        m_SelPts.RemoveAll();
        m_bAllowMouseRotate=m_bOldAllowRotate;
        Invalidate();
}

void CGLSelectableScatterGraph::OnLButtonDown(UINT nFlags, CPoint point)
{
        if (m_bMakeSel)
        {
                if (m_SelPts.GetSize()!=0)
                {
                        m_SelPts.RemoveAll();
//                      m_SelList.RemoveAll();
                        InvalidateRect(NULL);
                }
                m_SelPts.Add(point);
                m_PrevPt=point;
        }

        CGLScatterGraph::OnLButtonDown(nFlags, point);
}

void CGLSelectableScatterGraph::OnLButtonUp(UINT nFlags, CPoint point)
{
        if (m_bMakeSel==TRUE && m_SelPts.GetSize()>0)
        {
                CClientDC dc( this );
                dc.SelectStockObject(WHITE_PEN);
                dc.MoveTo( m_PrevPt );
                dc.LineTo( point );
                dc.LineTo(m_SelPts.GetAt(0));
                m_SelPts.Add(point);

                m_bMakeSel=FALSE;

//              MakeSelList();
//              GetParent()->SendMessage(DONE_SELECTING);
        }

        CGLScatterGraph::OnLButtonUp(nFlags, point);
}

void CGLSelectableScatterGraph::OnMouseMove(UINT nFlags, CPoint point)
{
// m_bMakeSel is set true by button click, and Capture is set
// in OnLButtonDown, which starts the drawing of the selection circle
```

```
        if (m_bMakeSel && m_SelPts.GetSize()>0) // check whether had a mouse down
        {
                CClientDC dc( this );

                dc.SelectStockObject(WHITE_PEN);
                m_SelPts.Add(point);
                // Draw a line from the previous detected point in the mouse
                // drag to the current point.
                dc.MoveTo( m_PrevPt );
                dc.LineTo( point );
                m_PrevPt = point;
        }
        CGLScatterGraph::OnMouseMove(nFlags, point);
}

void CGLSelectableScatterGraph::OnDrawGDI(CPaintDC *pDC)
{
TRACE("In CGLSelectableScatterGraph::OnDrawGDI\n");
        int i;
// draw the selection circle
        pDC->SelectStockObject(WHITE_PEN);
        int count=m_SelPts.GetSize();
        if (count>2)
        {
                pDC->MoveTo(m_SelPts.GetAt(0));
                for (i=1;i<count; i++)
                        pDC->LineTo(m_SelPts.GetAt(i));
                pDC->LineTo(m_SelPts.GetAt(0));
        }
#if 0
        for (i=0; i<test_num; i++)
        {
                pDC->SetPixelV(test_coords[i][0],test_coords[i][1],RGB(255,255,255));
        }
#endif
}

BOOL CGLSelectableScatterGraph::ZoomSel()
{
TRACE("entering ZoomSel\n");
#ifdef _DEBUG
        int errNum;
#endif
        int i,j,count;
        int id=0;
        GLint rv;
        GLfloat token;
        GLfloat *pBuf;
#ifdef _DEBUG
                int hitCount=0;
#endif
        if (m_SelPts.GetSize()>2)
        {
// make a region from selPts
                CPoint *pt=new CPoint[m_SelPts.GetSize()];
                for (i=0; i<m_SelPts.GetSize(); i++)
                        pt[i]=m_SelPts.GetAt(i);
                CRgn rgn;
                VERIFY(rgn.CreatePolygonRgn(pt,m_SelPts.GetSize(),ALTERNATE));
                delete [] pt;
// get a list of coordinate using feedback mode
// don't know how to calculate how much memory needed for buffer, but
// experiment shows the drawing has 96 floats overhead, + 4 per point
// give some spare, in case miscalculated !!
                pBuf=new GLfloat[200+m_Count*4];

                BeginGLCommands();
                glFeedbackBuffer(200+m_Count*4,GL_3D,pBuf);
                glRenderMode(GL_FEEDBACK);
#ifdef _DEBUG
                errNum=glGetError();
                TRACE("error on setting render mode = %s\n",gluErrorString(errNum));
                {
                        int rvRenderMode;
                        glGetIntegerv(GL_RENDER_MODE,&rvRenderMode);
                        CString mode;
                        if (rvRenderMode==GL_RENDER)
                                mode="Render";
                        else if (rvRenderMode==GL_FEEDBACK)
                                mode="Feedback";
                        else
                                mode="Unknown";
                TRACE("Render mode = %s\n",mode);
                }
#endif
                EndGLCommands();
TRACE("ZoomSel just about to update window to store coords\n");
                Invalidate();
                UpdateWindow();

                BeginGLCommands();
                rv=glRenderMode(GL_RENDER);
TRACE("return val from setting RenderMode = %d\n",rv);

/////////
// now find which data points have which screen coords
                int *pInSel=new int[m_Count];        // for each pt, will be 1 if in, 0 if out

                GLint viewport[4];
//              GLdouble mvmatrix[16],projmatrix[16];
//              GLdouble wx,wy,wz;
                glGetIntegerv(GL_VIEWPORT,viewport);
//              glGetDoublev(GL_MODELVIEW_MATRIX,mvmatrix);
//              glGetDoublev(GL_PROJECTION_MATRIX,projmatrix);
                EndGLCommands();
                count=rv;
                while (count)
                {
                        token=pBuf[rv-count];
                        count--;
                        if (token==GL_POINT_TOKEN)
                        {
//                              TRACE("Point token\n");
                                GLdouble coords[3];
                                for (j=0; j<3; j++)
                                {
//                                      TRACE("%4.2f ",pBuf[rv-count]);
                                        coords[j]=pBuf[rv-count];
                                        count--;
                                }
//                              TRACE("\n");
//                              gluUnProject(coords[0],coords[1],coords[2],
//                                      mvmatrix,projmatrix,viewport,
//                                      &wx,&wy,&wz);
//                              TRACE("render coords = %lf, %lf, %lf; realworld coords =
%lf, %lf, %lf\n",coords[0],coords[1],coords[2],wx,wy,wz);
                                CPoint pt;
                                pt.x=int(coords[0]);
                                pt.y=int(viewport[3]-coords[1]-1);
                                pInSel[id++]=rgn.PtInRegion(pt);
#ifdef _DEBUG
                                TRACE("pt %d has hit=%d\n",id-1,pInSel[id-1]);
                                if (rgn.PtInRegion(pt))
                                        hitCount++;
#endif
                        }
                }
                m_bAutoScaleX=m_bAutoScaleY=m_bAutoScaleZ=FALSE;
// find max & min
                float xMax,yMax,zMax,xMin,yMin,zMin;
                xMax=yMax=zMax=-FLT_MAX;
                xMin=yMin=zMin=FLT_MAX;
// drawCount is index of points which are drawn within old axes,
// and therefore appear in pSelList
                int drawCount=-1;
                for (i=0; i<m_Count; i++)
                {
                        if (!PtWithinAxes(m_pDat[i*3],m_pDat[i*3+1],m_pDat[i*3+2]))
                                continue;
                        drawCount++;
                        if (pInSel[drawCount]==0)
                                continue;
                        xMax=max(m_pDat[i*3],xMax);
                        xMin=min(m_pDat[i*3],xMin);
                        yMax=max(m_pDat[i*3+1],yMax);
                        yMin=min(m_pDat[i*3+1],yMin);
                        zMax=max(m_pDat[i*3+2],zMax);
                        zMin=min(m_pDat[i*3+2],zMin);
                }
                m_MaxX=NextAbove(xMax,5);
                m_MinX=NextBelow(xMin,5);
                m_MaxY=NextAbove(yMax,5);
                m_MinY=NextBelow(yMin,5);
                m_MaxZ=NextAbove(zMax,5);
                m_MinZ=NextBelow(zMin,5);
                Invalidate();
                delete [] pBuf;
                delete [] pInSel;
                rgn.DeleteObject();
        }
        m_bAllowMouseRotate=m_bOldAllowRotate;
        CancelSel();

#ifdef _DEBUG
TRACE("leaving ZoomSel, %d points on screen, %d in selection\n",id,hitCount);
#endif

        return TRUE;
}
```

## OpenGLWnd.cpp

```
// OpenGLWnd.cpp : implementation file
//
/* SGI openGL libraries (link with OPENGL.LIB and GLU.LIB)
#include "[path-of-SGI-sdk]\include\gl\gl.h"
#include "[path-of-SGI-sdk]\include\gl\glu.h"
//*/
//#include "afxtempl.h"
#include "stdafx.h"
#include "OpenGLWnd.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


#define MAX_LISTS 20
// used to identify a MCD video driver (partial OGL acceleration)
#define INSTALLABLE_DRIVER_TYPE_MASK
(PFD_GENERIC_ACCELERATED|PFD_GENERIC_FORMAT)

/////////////////////////////////////////////////////////////////////////////
// COpenGLWnd

COpenGLWnd::COpenGLWnd() :
        m_dAspectRatio(1.0),
        m_bInsideDispList(FALSE), m_bExternDispListCall(FALSE),
        m_bExternGLCall(FALSE)

{
// define a default cursor
        m_hMouseCursor=AfxGetApp()->LoadStandardCursor(IDC_ARROW);
// set the disp list vector to all zeros
        for (int c=0;c<MAX_LISTS;c++) m_DispListVector[c]=0;

        m_FontListBase=1000;      // initial guess for font display list
        m_bGotFont=FALSE;
}

COpenGLWnd::~COpenGLWnd()
{
}


BEGIN_MESSAGE_MAP(COpenGLWnd, CWnd)
        //{{AFX_MSG_MAP(COpenGLWnd)
        ON_WM_CREATE()
        ON_WM_DESTROY()
        ON_WM_ERASEBKGND()
        ON_WM_SIZE()
        ON_WM_SETCURSOR()
        ON_WM_PAINT()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////
// COpenGLWnd Constants

// these are used to construct an equilibrated 256 color palette
static unsigned char _threeto8[8] =
{
        0, 0111>>1, 0222>>1, 0333>>1, 0444>>1, 0555>>1, 0666>>1, 0377
};

static unsigned char _twoto8[4] =
{
        0, 0x55, 0xaa, 0xff
};

static unsigned char _oneto8[2] =
{
        0, 255
};

static int defaultOverride[13] =
{
        0, 3, 24, 27, 64, 67, 88, 173, 181, 236, 247, 164, 91
};

// Windows Default Palette
static PALETTEENTRY defaultPalEntry[20] =
{
        { 0,   0,   0,    0 },
        { 0x80,0,   0,    0 },
        { 0,   0x80,0,    0 },
        { 0x80,0x80,0,    0 },
        { 0,   0,   0x80, 0 },
        { 0x80,0,   0x80, 0 },
        { 0,   0x80,0x80, 0 },
        { 0xC0,0xC0,0xC0, 0 },

        { 192, 220, 192,  0 },
        { 166, 202, 240,  0 },
        { 255, 251, 240,  0 },
        { 160, 160, 164,  0 },

        { 0x80,0x80,0x80, 0 },
        { 0xFF,0,   0,    0 },
        { 0,   0xFF,0,    0 },
        { 0xFF,0xFF,0,    0 },
        { 0,   0,   0xFF, 0 },
        { 0xFF,0,   0xFF, 0 },
        { 0,   0xFF,0xFF, 0 },
        { 0xFF,0xFF,0xFF, 0 }
};


/////////////////////////////////////////////////////////////////////////////
// COpenGLWnd message handlers

int COpenGLWnd::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

// OpenGL rendering context creation
        PIXELFORMATDESCRIPTOR pfd;
   int       n;

// initialize the private member
        m_pCDC= new CClientDC(this);

// choose the requested video mode
   if (!bSetupPixelFormat()) return 0;

// ask the system if the video mode is supported
   n=::GetPixelFormat(m_pCDC->GetSafeHdc());
   ::DescribePixelFormat(m_pCDC->GetSafeHdc(),n,sizeof(pfd),&pfd);

// create a palette if the requested video mode has 256 colors (indexed mode)
   CreateRGBPalette();

// link the Win Device Context with the OGL Rendering Context
```

```
   m_hRC = wglCreateContext(m_pCDC->GetSafeHdc());

// specify the target DeviceContext (window) of the subsequent OGL calls
   wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);

// performs default setting of rendering mode,etc..
        OnCreateGL();

// free the target DeviceContext (window)
   wglMakeCurrent(NULL,NULL);

        return 0;
}

void COpenGLWnd::OnDestroy()
{
// specify the target DeviceContext (window) of the subsequent OGL calls
   wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);

// remove all display lists
        for (int c=0;c<MAX_LISTS;c++) if(m_DispListVector[c])
glDeleteLists(m_DispListVector[c],1);

// release definitely OGL Rendering Context
        if (m_hRC!=NULL) ::wglDeleteContext(m_hRC);

// Select our palette out of the dc
        CPalette palDefault;
        palDefault.CreateStockObject(DEFAULT_PALETTE);
        m_pCDC->SelectPalette(&palDefault, FALSE);

// destroy Win Device Context
        if(m_pCDC) delete m_pCDC;

// finally call the base function
        CWnd::OnDestroy();
}

BOOL COpenGLWnd::OnEraseBkgnd(CDC* pDC)
{
// OGL has his own background erasing so tell Windows to skip
        return TRUE;
}

void COpenGLWnd::OnSize(UINT nType, int cx, int cy)
{
	CWnd::OnSize(nType, cx, cy);

// when called with a nonzero window:
	if ( 0 < cx && 0 < cy )
	{
// update the rect and the aspect ratio
		m_ClientRect.right = cx;
		m_ClientRect.bottom = cy;
		m_dAspectRatio=double(cx)/double(cy);

// specify the target DeviceContext of the subsequent OGL calls
		wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);

// call the virtual sizing procedure (to be overridden by user)
		OnSizeGL(cx,cy);

// free the target DeviceContext (window)
		wglMakeCurrent(NULL,NULL);

// force redraw
		Invalidate(TRUE);
	};
}

// NOTE: this does not work if a derived class captures the mouse.
// The cursor must then be set explicitly with each Mouse call.
BOOL COpenGLWnd::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
	ASSERT(m_hMouseCursor!=NULL);
	::SetCursor(m_hMouseCursor);

	return TRUE;
}

// pass in origin (bottom,left,near), width, height, depth
// NOTE, to make box go into screen from org, z must be negative
void COpenGLWnd::DrawBox(CPoint3D org, float x, float y, float z)
{
	glBegin( GL_LINE_LOOP );
		glVertex3f(org.x, org.y, org.z);
		glVertex3f(org.x, org.y+y, org.z);
		glVertex3f(org.x+x, org.y+y, org.z);
		glVertex3f(org.x+x, org.y, org.z);
		glVertex3f(org.x+x, org.y, org.z+z);
		glVertex3f(org.x+x, org.y+y, org.z+z);
		glVertex3f(org.x, org.y+y, org.z+z);
```

```
                glVertex3f(org.x, org.y, org.z+z);
        glEnd();
        glBegin(GL_LINES);
                glVertex3f(org.x, org.y, org.z);
                glVertex3f(org.x+x, org.y, org.z);
                glVertex3f(org.x, org.y, org.z+z);
                glVertex3f(org.x+x, org.y, org.z+z);

                glVertex3f(org.x, org.y+y, org.z);
                glVertex3f(org.x, org.y+y, org.z+z);
                glVertex3f(org.x+x, org.y+y, org.z);
                glVertex3f(org.x+x, org.y+y, org.z+z);
        glEnd();
}


void COpenGLWnd::OnPaint()
{
TRACE("In COpenGLWnd::OnPaint\n");
// prepare a semaphore
        static BOOL   bBusy = FALSE;
// use the semaphore to enter this critic section
        if(bBusy) return;
        bBusy = TRUE;

// specify the target DeviceContext of the subsequent OGL calls
//    wglMakeCurrent(dc.m_ps.hdc, m_hRC);
        wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);

// clear background
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

#ifdef _DEBUG
        int rvRenderMode;
        glGetIntegerv(GL_RENDER_MODE,&rvRenderMode);
        CString mode;
        if (rvRenderMode==GL_RENDER)
                mode="Render";
        else if (rvRenderMode==GL_FEEDBACK)
                mode="Feedback";
        else
                mode="Unknown";
TRACE("In COpenGLWnd::OnPaint, mode = %s\n",mode);
#endif
```

```
// call the virtual drawing procedure (to be overridden by user)
        OnDrawGL();

// execute OGL commands (flush the OGL graphical pipeline)
        glFinish();

// if double buffering is used it's time to swap the buffers
//      SwapBuffers(dc.m_ps.hdc);
        SwapBuffers(m_pCDC->GetSafeHdc());

// turn the semaphore "green"
        bBusy = FALSE;

// free the target DeviceContext (window)
   wglMakeCurrent(NULL,NULL);

// do any GDI drawing
        CPaintDC dc(this); // device context for painting
        OnDrawGDI(&dc);
}

/////////////////////////////////////////////////////////////////////////////
// COpenGLWnd public members

void COpenGLWnd::VideoMode(ColorsNumber &c, ZAccuracy &z, BOOL &dbuf)
{
// set default videomode
        c=MILLIONS;
        z=NORMAL;
        dbuf=TRUE;
}

void COpenGLWnd::SetMouseCursor(HCURSOR mcursor)
{
// set the specified cursor (only if it is a valid one)
        if(mcursor!=NULL) m_hMouseCursor=mcursor;
}

const CString COpenGLWnd::GetInformation(InfoField type)
{
        PIXELFORMATDESCRIPTOR pfd;
        CString str("Not Available");

// Get information about the DC's current pixel format
        ::DescribePixelFormat( m_pCDC->GetSafeHdc(), ::GetPixelFormat(m_pCDC-
>GetSafeHdc()),sizeof(PIXELFORMATDESCRIPTOR), &pfd );
```

```
// specify the target DeviceContext of the subsequent OGL calls
        wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);

        switch(type)
        {
           // Derive driver information
        case ACCELERATION: if( 0==(INSTALLABLE_DRIVER_TYPE_MASK &
        pfd.dwFlags) ) str="Fully        Accelerated (ICD)"; // fully in hardware (fastest)
        else if
(INSTALLABLE_DRIVER_TYPE_MASK==(INSTALLABLE_DRIVER_TYPE_MASK
& pfd.dwFlags) ) str="Partially Accelerated (MCD)"; // partially in hardware (pretty fast,
maybe..)
                 else str="Not Accelerated (Software)";        // software
                                                  break;
// get the company name responsible for this implementation
        case VENDOR:str=(char*)::glGetString(GL_VENDOR);
                if ( ::glGetError()!=GL_NO_ERROR) str.Format("Not Available");// failed!
                                break;
// get the renderer name; this is specific of an hardware configuration
        case RENDERER:str=(char*)::glGetString(GL_RENDERER);
                if ( ::glGetError()!=GL_NO_ERROR) str.Format("Not Available");// failed!
                                        break;
// get the version
        case VERSION:str=(char*)::glGetString(GL_VERSION);
                if ( ::glGetError()!=GL_NO_ERROR) str.Format("Not Available");// failed!
                                break;
// return a space separated list of extensions
        case EXTENSIONS: str=(char*)::glGetString(GL_EXTENSIONS);
                if ( ::glGetError()!=GL_NO_ERROR) str.Format("Not Available");// failed!
                                break;
        };

// free the target DeviceContext (window) and return the result
        wglMakeCurrent(NULL,NULL);
        return str;
}

void COpenGLWnd::DrawStockDispLists()
{
// check if we are already inside a drawing session
        if(m_hRC==wglGetCurrentContext() && m_pCDC-
>GetSafeHdc()==wglGetCurrentDC() )
        {
// draw directly all display lists
```

```
                for (int c=0;c<MAX_LISTS;c++) if(m_DispListVector[c])
glCallList(m_DispListVector[c]);
        }
        else
        {
// specify the target DeviceContext of the subsequent OGL calls
                wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);
// draw all display lists
                for (int c=0;c<MAX_LISTS;c++) if(m_DispListVector[c])
glCallList(m_DispListVector[c]);
// free the target DeviceContext (window)
                wglMakeCurrent(NULL,NULL);
        };
}

void COpenGLWnd::StartStockDListDef()
{
// check if we aren't inside another couple begin/end
        if(!m_bInsideDispList)
        {
// search a free slot
                for (int c=0;m_DispListVector[c]!=0;c++);
// check if we are inside a drawing session or not....
                if(!( m_hRC==wglGetCurrentContext() && m_pCDC-
>GetSafeHdc()==wglGetCurrentDC() ))
                {
// ...if not specify the target DeviceContext of the subsequent OGL calls
                        wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);
// set a warning for EndDispList
                        m_bExternDispListCall=TRUE;
                };
// create a handle to the disp list (actually an integer)
                m_DispListVector[c]=glGenLists(1);
// set a semaphore
                m_bInsideDispList=TRUE;
// start the disp list: all subsequent OGL calls will be redirected to the list
                glNewList(m_DispListVector[c],GL_COMPILE);
        };
}

void COpenGLWnd::EndStockListDef()
{
// close the disp list
        glEndList();
// unset the semaphore
        m_bInsideDispList=FALSE;
```

```
// if beginDispList set the warn free the target DeviceContext
        if(m_bExternDispListCall) wglMakeCurrent(NULL,NULL);
}

void COpenGLWnd::ClearStockDispLists()
{
// check if we are referring to the right Rendering Context
        if(m_hRC==wglGetCurrentContext() && m_pCDC-
>GetSafeHdc()==wglGetCurrentDC() )
        {
// delete active display lists
                for (int c=0;c<MAX_LISTS;c++) if(m_DispListVector[c])
glDeleteLists(m_DispListVector[c],1);
        }
        else
        {
// specify the target Rendering Context of the subsequent OGL calls
                wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);
// delete active display lists
                for (int c=0;c<MAX_LISTS;c++) if(m_DispListVector[c])
glDeleteLists(m_DispListVector[c],1);
// free the target Rendering Context (window)
                wglMakeCurrent(NULL,NULL);
        };
}

void COpenGLWnd::BeginGLCommands()
{
// check if we are inside a drawing session or not....
        if(!( m_hRC==wglGetCurrentContext() && m_pCDC-
>GetSafeHdc()==wglGetCurrentDC() ))
        {
// ...if not specify the target DeviceContext of the subsequent OGL calls
                wglMakeCurrent(m_pCDC->GetSafeHdc(), m_hRC);
// set a warning for EndGLCommands
                m_bExternGLCall=TRUE;
        };
}

void COpenGLWnd::EndGLCommands()
{
// if BeginGLCommands set the warn free the target DeviceContext
        if(m_bExternGLCall) wglMakeCurrent(NULL,NULL);
}

//////////////////////////////////////////////////////////////////////
```

```
//
// Implementation of COpenGLWnd::CGLDispList class.
//
//////////////////////////////////////////////////////////////////////
// Construction/Destruction

COpenGLWnd::CGLDispList::CGLDispList():
        m_glListId(0), m_bIsolated(FALSE)
{
}

COpenGLWnd::CGLDispList::~CGLDispList()
{
// remove display list
        glDeleteLists(m_glListId,1);
}

//////////////////////////////////////////////////////////////////////
// Member functions

void COpenGLWnd::CGLDispList::Draw()
{
// if the list is not empty...
        if(m_glListId)
        {
                if(m_bIsolated)
                {
// save current transformation matrix
                        glPushMatrix();
// save current OGL internal state (lighting, shading, and such)
                        glPushAttrib(GL_ALL_ATTRIB_BITS);
                };
// draw the list
                glCallList(m_glListId);
                if(m_bIsolated)
                {
// restore transformation matrix
                        glPopMatrix();
// restore OGL internal state
                        glPopAttrib();
                };
        };
}

void COpenGLWnd::CGLDispList::StartDef(BOOL bImmediateExec)
{
```

```
// set the context for GL calls (if needed)
//        BeginGLCommands();
// check if another list is under construction
          int cur;
          glGetIntegerv(GL_LIST_INDEX,&cur);
          if(cur != 0) {TRACE("Error: Nested display list definition!");ASSERT(FALSE);};
// if the list is empty firstly allocate one
          if(!m_glListId) m_glListId=glGenLists(1);

// start or replace a list definition
          if (bImmediateExec) glNewList(m_glListId,GL_COMPILE_AND_EXECUTE);
          else  glNewList(m_glListId,GL_COMPILE);
}

void COpenGLWnd::CGLDispList::EndDef()
{
// check the coupling with a preceding call to StartDef()
          int cur;
          glGetIntegerv(GL_LIST_INDEX,&cur);
          if(cur != m_glListId) {TRACE("CGLDispList:Missing StartDef() before
EndDef()\n");return;};
// close list definition
          glEndList();
// free the context (if needed)
//        EndGLCommands();
}

void COpenGLWnd::CreateRGBPalette()
{
   PIXELFORMATDESCRIPTOR pfd;
   LOGPALETTE *pPal;
   int n, i;

// get the initially choosen video mode
          n = ::GetPixelFormat(m_pCDC->GetSafeHdc());
   ::DescribePixelFormat(m_pCDC->GetSafeHdc(), n, sizeof(pfd), &pfd);

// if is an indexed one...
   if (pfd.dwFlags & PFD_NEED_PALETTE)
   {
// ... construct an equilibrated palette (3 red bits, 3 green bits, 2 blue bits)
// NOTE: this code is integrally taken from MFC example Cube
                  n = 1 << pfd.cColorBits;
      pPal = (PLOGPALETTE) new char[sizeof(LOGPALETTE) + n *
sizeof(PALETTEENTRY)];
```

```
      ASSERT(pPal != NULL);

      pPal->palVersion = 0x300;
      pPal->palNumEntries = n;
      for (i=0; i<n; i++)
      {
         pPal->palPalEntry[i].peRed=ComponentFromIndex(i, pfd.cRedBits,
pfd.cRedShift);
         pPal->palPalEntry[i].peGreen=ComponentFromIndex(i, pfd.cGreenBits,
pfd.cGreenShift);
         pPal->palPalEntry[i].peBlue=ComponentFromIndex(i, pfd.cBlueBits,
pfd.cBlueShift);
         pPal->palPalEntry[i].peFlags=0;
      }

// fix up the palette to include the default Windows palette
      if ((pfd.cColorBits == 8)                          &&
         (pfd.cRedBits   == 3) && (pfd.cRedShift   == 0) &&
         (pfd.cGreenBits == 3) && (pfd.cGreenShift == 3) &&
         (pfd.cBlueBits  == 2) && (pfd.cBlueShift  == 6)
        )
      {
                        for (i = 1 ; i <= 12 ; i++)
            pPal->palPalEntry[defaultOverride[i]] = defaultPalEntry[i];
      }

      m_CurrentPalette.CreatePalette(pPal);
      delete pPal;

// set the palette
      m_pOldPalette=m_pCDC->SelectPalette(&m_CurrentPalette, FALSE);
      m_pCDC->RealizePalette();
   }
}

unsigned char COpenGLWnd::ComponentFromIndex(int i, UINT nbits, UINT shift)
{
   unsigned char val;

   val = (unsigned char) (i >> shift);
   switch (nbits)
         {

   case 1:
      val &= 0x1;
      return _oneto8[val];
```

```
  case 2:
    val &= 0x3;
    return _twoto8[val];
  case 3:
    val &= 0x7;
    return _threeto8[val];

  default:
    return 0;
  }
}

BOOL COpenGLWnd::bSetupPixelFormat()
{
// define default desired video mode (pixel format)
       static PIXELFORMATDESCRIPTOR pfd =
       {
    sizeof(PIXELFORMATDESCRIPTOR),          // size of this pfd
    1,                                             // version number
    PFD_DRAW_TO_WINDOW |                    // support window
    PFD_SUPPORT_OPENGL |                    // support OpenGL
    PFD_DOUBLEBUFFER,                              // double buffered
    PFD_TYPE_RGBA,           // RGBA type
    24,                  // 24-bit color depth
    0, 0, 0, 0, 0, 0,      // color bits ignored
    0,                   // no alpha buffer
    0,                   // shift bit ignored
    0,                   // no accumulation buffer
    0, 0, 0, 0,            // accum bits ignored
    16,                  // 32-bit z-buffer
    0,                   // no stencil buffer
    0,                   // no auxiliary buffer
    PFD_MAIN_PLANE,          // main layer
    0,                   // reserved
    0, 0, 0               // layer masks ignored
  };
// let the user change some parameters if he wants
       BOOL bDoublBuf;
       ColorsNumber cnum;
       ZAccuracy zdepth;
       VideoMode(cnum,zdepth,bDoublBuf);
//set the changes
       if(bDoublBuf) pfd.dwFlags=PFD_DRAW_TO_WINDOW |
PFD_SUPPORT_OPENGL |PFD_DOUBLEBUFFER;
       else pfd.dwFlags=PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL;
       switch(cnum)
```

```
      {
      case INDEXED: pfd.cColorBits=8;
      case THOUSANDS: pfd.cColorBits=16;
      case MILLIONS: pfd.cColorBits=24;
      case MILLIONS_WITH_TRANSPARENCY: pfd.cColorBits=32;
      };
      switch(zdepth)
      {
      case NORMAL: pfd.cDepthBits=16;
      case ACCURATE: pfd.cDepthBits=32;
      };

// ask the system for such video mode
  ASSERT(m_pCDC != NULL);
  int pixelformat;
      if ( (pixelformat = ChoosePixelFormat(m_pCDC->GetSafeHdc(), &pfd)) == 0 )
  {
    AfxMessageBox("ChoosePixelFormat failed");
    return FALSE;
  }

// try to set this video mode
      if (SetPixelFormat(m_pCDC->GetSafeHdc(), pixelformat, &pfd) == FALSE)
  {
// the requested video mode is not available so get a default one
    pixelformat = 1;
            if (DescribePixelFormat(m_pCDC->GetSafeHdc(), pixelformat,
sizeof(PIXELFORMATDESCRIPTOR), &pfd)==0)
            {
// neither the requested nor the default are available: fail
                  AfxMessageBox("SetPixelFormat failed (no OpenGL compatible
video mode)");
                  return FALSE;
            }
  }

  return TRUE;
}


void COpenGLWnd::OnCreateGL()
{
// perform hidden line/surface removal (enabling Z-Buffer)
      glEnable(GL_DEPTH_TEST);

// set background color to black
```

```
        glClearColor(0.f,0.f,0.f,1.0f );

// set clear Z-Buffer value
        glClearDepth(1.0f);
}


void COpenGLWnd::OnDrawGL()
{
TRACE("In COpenGLWnd::OnDrawGL\n");
// draw carthesian axes
        glBegin(GL_LINES);
                // red x axis
                glColor3f(1.f,0.f,0.f);
                glVertex3f(0.0f,0.0f,0.0f);
                glVertex3f(1.0f,0.0f,0.0f);
                glVertex3f(1.0f,0.0f,0.0f);
                glVertex3f(0.9f,0.1f,0.0f);
                glVertex3f(1.0f,0.0f,0.0f);
                glVertex3f(0.9f,-0.1f,0.0f);
                // green y axis
                glColor3f(0.f,1.f,0.f);
                glVertex3f(0.0f,0.0f,0.0f);
                glVertex3f(0.0f,1.0f,0.0f);
                glVertex3f(0.0f,1.0f,0.0f);
                glVertex3f(0.1f,0.9f,0.0f);
                glVertex3f(0.0f,1.0f,0.0f);
                glVertex3f(-0.1f,0.9f,0.0f);
                // blue z axis
                glColor3f(0.f,0.f,1.f);
                glVertex3f(0.0f,0.0f,0.0f);
                glVertex3f(0.0f,0.0f,1.0f);
                glVertex3f(0.0f,0.0f,1.0f);
                glVertex3f(0.0f,0.1f,0.9f);
                glVertex3f(0.0f,0.0f,1.0f);
                glVertex3f(0.0f,-0.1f,0.9f);
        glEnd();
}
void COpenGLWnd::OnDrawGDI(CPaintDC *pDC)
{
TRACE("In COpenGLWnd::OnDrawGDI\n");
}

void COpenGLWnd::OnSizeGL(int cx, int cy)
{
// set correspondence between window and OGL viewport
```

```
        glViewport(0,0,cx,cy);

// update the camera
        glPushMatrix();
            glMatrixMode(GL_PROJECTION);
                glLoadIdentity();
                gluPerspective(40.0,m_dAspectRatio,0.1f, 10.0f);
                glTranslatef(0.0f,0.0f,-4.f);
            glMatrixMode(GL_MODELVIEW);
        glPopMatrix();
}

void COpenGLWnd::SetClearCol(COLORREF rgb)
{
    float r=float(GetRValue(rgb))/255;
    float g=float(GetGValue(rgb))/255;
    float b=float(GetBValue(rgb))/255;
    BeginGLCommands();
    glClearColor(r,g,b,1.0f);
    EndGLCommands();
    Invalidate();   // force redraw
}

void COpenGLWnd::MakeFont()
{
    int i;
    BeginGLCommands();
// check m_FontListBase not in use
    BOOL bUsed=FALSE;
    do
    {
        for (i=0; i<255; i++)
        {
            if (glIsList(m_FontListBase+i))
            {
                m_FontListBase+=256;
                bUsed=TRUE;
                break;
            }
        }
    } while (bUsed==TRUE);

    SelectObject (m_pCDC->GetSafeHdc(), GetStockObject (SYSTEM_FONT));
// create the bitmap display lists
// we're making images of glyphs 0 thru 255
// the display list numbering starts at m_FontListBase, an arbitrary choice
```

```
        m_bGotFont=wglUseFontBitmaps (m_pCDC->GetSafeHdc(), 0, 255,
m_FontListBase);
        EndGLCommands();
}

void COpenGLWnd::PrintString(const char* str)
{
        if (!m_bGotFont)
                return;
     glPushAttrib(GL_LIST_BIT);
     glListBase(m_FontListBase);
     glCallLists(strlen(str), GL_UNSIGNED_BYTE, (GLubyte*)str);
     glPopAttrib();
}

void COpenGLWnd::CopyToClipboard()
{
        CRect R;
        GetWindowRect(&R);
        if (!OpenClipboard())
        {
#ifdef _DEBUG
                TRACE("Cannot open clipboard.\n");
#endif
        return;
        }
        CWindowDC dc(this);
        CBitmap* pbmOld = NULL;
        CDC dcMem;
        VERIFY(dcMem.CreateCompatibleDC(&dc));

        CBitmap bm;
        VERIFY(bm.CreateCompatibleBitmap(&dc,R.Width(),R.Height()));
        ASSERT(bm.m_hObject != NULL);
        pbmOld = dcMem.SelectObject(&bm);

        dcMem.PatBlt(0,0,R.Width(),R.Height(),WHITENESS);
        VERIFY(dcMem.BitBlt(0,0,R.Width(),R.Height(),
                &dc,0,0,SRCCOPY));
        HGDIOBJ hBM=bm.Detach();
        VERIFY(::EmptyClipboard());
        VERIFY(::SetClipboardData(CF_BITMAP, hBM));
        VERIFY(::CloseClipboard());

        dcMem.SelectObject(pbmOld);
        dcMem.DeleteDC();}
```

## Scatter3D.cpp

```
#include "stdafx.h"
#include "Scatter3D.h"
#include "Scatter3DDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
// CScatter3DApp

BEGIN_MESSAGE_MAP(CScatter3DApp, CWinApp)
        //{{AFX_MSG_MAP(CScatter3DApp)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
// CScatter3DApp construction

CScatter3DApp::CScatter3DApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}
// The one and only CScatter3DApp object
CScatter3DApp theApp;
// CScatter3DApp initialization

BOOL CScatter3DApp::InitInstance()
{
        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.
#ifdef _AFXDLL
        Enable3dControls();                     // Call this when using MFC in a shared DLL
#else
        Enable3dControlsStatic();       // Call this when linking to MFC statically
#endif
        CScatter3DDlg dlg;
        m_pMainWnd = &dlg;
        int nResponse = dlg.DoModal();
        if (nResponse == IDOK)
        {
                // TODO: Place code here to handle when the dialog is
                //  dismissed with OK
        }
        else if (nResponse == IDCANCEL)
        {
                // TODO: Place code here to handle when the dialog is
                //  dismissed with Cancel
        }
        // Since the dialog has been closed, return FALSE so that we exit the
        //  application, rather than start the application's message pump.
        return FALSE;
}
```

### Scatter3DDlg.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "stdafx.h"
#include "Scatter3D.h"
#include "Scatter3DDlg.h"
#include "GLSelectableScatterGraph.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#define STRLEN 1000

CString file;

CScatter3DDlg::CScatter3DDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CScatter3DDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CScatter3DDlg)
        m_ProjType = 0;
        m_SymbolSize = 5;
        m_bMouseRotate = TRUE;
        m_MaxX = 1.0f;
        m_MaxY = 1.0f;
        m_MaxZ = 1.0f;
        m_MinX = 0.0f;
        m_MinY = 0.0f;
        m_MinZ = 0.0f;
        m_bAutoX = TRUE;
        m_bAutoY = TRUE;
        m_bAutoZ = TRUE;
        m_DatCount = 4;
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
        m_pDisplay = new CGLSelectableScatterGraph;
        m_ClearCol=RGB(0,0,0);
        m_PtCol=RGB(255,0,0);

        m_pData=NULL;
        m_pColList=NULL;
```

```
}

void CScatter3DDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CScatter3DDlg)
        DDX_Control(pDX, IDC_COMBO1, m_combox);
        DDX_Control(pDX, IDC_COMBO2, m_comboy);
   DDX_Control(pDX, IDC_COMBO3, m_comboz);
        DDX_Radio(pDX, IDC_PROJ_TYPE, m_ProjType);
        DDX_Text(pDX, IDC_SYMBOL_SIZE, m_SymbolSize);
        DDV_MinMaxInt(pDX, m_SymbolSize, 1, 20);
        DDX_Check(pDX, IDC_MOUSE_ROTATE, m_bMouseRotate);
        DDX_Text(pDX, IDC_MAX_X, m_MaxX);
        DDX_Text(pDX, IDC_MAX_Y, m_MaxY);
        DDX_Text(pDX, IDC_MAX_Z, m_MaxZ);
        DDX_Text(pDX, IDC_MIN_X, m_MinX);
        DDX_Text(pDX, IDC_MIN_Y, m_MinY);
        DDX_Text(pDX, IDC_MIN_Z, m_MinZ);
        DDX_Check(pDX, IDC_AUTO_X, m_bAutoX);
        DDX_Check(pDX, IDC_AUTO_Y, m_bAutoY);
        DDX_Check(pDX, IDC_AUTO_Z, m_bAutoZ);
        DDX_Text(pDX, IDC_DAT_COUNT, m_DatCount);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CScatter3DDlg, CDialog)
        //{{AFX_MSG_MAP(CScatter3DDlg)
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_WM_SIZE()
        ON_BN_CLICKED(IDC_PROJ_TYPE, OnProjType)
        ON_BN_CLICKED(IDC_PROJ_TYPE2, OnProjType2)
        ON_BN_CLICKED(IDC_BACK_COLOUR, OnBackColour)
        ON_WM_DRAWITEM()
        ON_EN_KILLFOCUS(IDC_SYMBOL_SIZE, OnKillfocusSymbolSize)
        ON_BN_CLICKED(IDC_MOUSE_ROTATE, OnMouseRotate)
        ON_BN_CLICKED(IDC_AUTO_X, OnAutoX)
        ON_BN_CLICKED(IDC_AUTO_Y, OnAutoY)
        ON_BN_CLICKED(IDC_AUTO_Z, OnAutoZ)
        ON_EN_KILLFOCUS(IDC_MAX_X, OnKillfocusMaxX)
        ON_EN_KILLFOCUS(IDC_MAX_Y, OnKillfocusMaxY)
        ON_EN_KILLFOCUS(IDC_MAX_Z, OnKillfocusMaxZ)
        ON_EN_KILLFOCUS(IDC_MIN_X, OnKillfocusMinX)
        ON_EN_KILLFOCUS(IDC_MIN_Y, OnKillfocusMinY)
        ON_EN_KILLFOCUS(IDC_MIN_Z, OnKillfocusMinZ)
        ON_BN_CLICKED(IDC_MAKE_SEL, OnMakeSel)
        ON_BN_CLICKED(IDC_CANCEL_SEL, OnCancelSel)
        ON_BN_CLICKED(IDC_ZOOM_SEL, OnZoomSel)
        ON_BN_CLICKED(IDC_COPY, OnCopy)
        ON_BN_CLICKED(IDC_LOAD, OnLoad)
        ON_WM_DROPFILES()
        ON_BN_CLICKED(IDC_PT_COLOUR, OnPtColour)
        ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScatter3DDlg message handlers

BOOL CScatter3DDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // Set the icon for this dialog.  The framework does this automatically
        //  when the application's main window is not a dialog
        SetIcon(m_hIcon, TRUE);                         // Set big icon
        SetIcon(m_hIcon, FALSE);                // Set small icon

        // Enable drag/drop open
        DragAcceptFiles();

        CRect rect(10,10,10,10);

        // TODO: Add extra initialization here
        m_pDisplay->Create( NULL,   //CWnd default
                                        NULL,   //has no name

        WS_CHILD|WS_CLIPSIBLINGS|WS_CLIPCHILDREN|WS_VISIBLE,
                                        rect,
                                        this,   //this is the parent
                                        0);     //this should really be a different
number... check resource.h

// generate OnSize to get Graph window positioned right
        CRect r;
        GetWindowRect(&r);
        r.InflateRect(1,1);
   MoveWindow(r);

        m_pDisplay->SetProjection(m_ProjType);
        m_pDisplay->SetSymbolSize(m_SymbolSize);
        m_pDisplay->SetClearCol(m_ClearCol);
        m_pDisplay->AllowMouseRotate(m_bMouseRotate);//0.5,    1.,        0.,

        static float fake_data[]= {
                      0.5,  0.5,  0.5,
                      0.5, -0.5,  0.5,
                     -0.5, -0.5, 0.5,
                          -0.5,  0.5,     0.5,
                     -0.5, -0.5,-0.5,
                      0.5, -0.5,-0.5,
                     -0.5,  0.5,-0.5,
                      0.5,  0.5, -0.5
                          };

        static COLORREF fake_colList[] = {
                          RGB(255,255,255),
                          RGB(255,255,255),
                          RGB(255,255,255),
                          RGB(255,255,255),
                          RGB(155,155,155),
                          RGB(155,155,155),
                          RGB(155,155,155),
                          RGB(155,155,155)

                              };

        m_pDisplay->SetData(8,RGB(255,0,0),fake_data,fake_colList);
        OnAutoX();    // scale data and fill edit boxes
        OnAutoY();
        OnAutoZ();

        return TRUE;  // return TRUE  unless you set the focus to a control
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CScatter3DDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);
```

```
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CScatter3DDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

CScatter3DDlg::~CScatter3DDlg()
{
    if(m_pDisplay)
    {
        delete m _pDisplay;
    }
    if (m_pData!=NULL)
        delete [] m_pData;
    if (m_pColList!=NULL)
        delete [] m_pColList;
}

void CScatter3DDlg::OnSize(UINT nType, int cx, int cy)
{
    CDialog::OnSize(nType, cx, cy);

    if (m_pDisplay->GetSafeHwnd()==NULL)            // OnSize is first called before
Buttons created??
        return;

    CRect R;
```

```
   GetDlgItem(IDC_GRAPH_TOPLEFT)->GetWindowRect(R);              // get
dimensions of Button
   ScreenToClient(R);
//   R.left=R.right+10;
//   R.top=10;
   R.right=cx-10;
   R.bottom=cy-10;
   m_pDisplay->MoveWindow(R);
}

void CScatter3DDlg::OnProjType()
{
         UpdateData();
         m_pDisplay->SetProjection(m_ProjType);
         m_pDisplay->Invalidate(FALSE);
}

void CScatter3DDlg::OnProjType2()
{
         UpdateData();
         m_pDisplay->SetProjection(m_ProjType);
         m_pDisplay->Invalidate(FALSE);
}

// Changing the background color
void CScatter3DDlg::OnBackColour()
{
         CColorDialog dlg;
         if (dlg.DoModal()!=IDOK)
                  return;

         m_ClearCol=dlg.GetColor(); // for drawing button
         GetDlgItem(IDC_BACK_COLOUR)->Invalidate();
         m_pDisplay->SetClearCol(m_ClearCol);
}

void CScatter3DDlg::OnDrawItem(int nIDCtl, LPDRAWITEMSTRUCT
lpDrawItemStruct)
{
         if (nIDCtl ==IDC_BACK_COLOUR || nIDCtl ==IDC_PT_COLOUR)
         {
                  if (lpDrawItemStruct->itemAction & ODA_DRAWENTIRE)
                  {
                           HBRUSH hBrush;
                           if (nIDCtl ==IDC_BACK_COLOUR)
                                    hBrush=::CreateSolidBrush(m_ClearCol);
```

```
                        else
                                hBrush=::CreateSolidBrush(m_PtCol);

                        ::FillRect(lpDrawItemStruct->hDC,&(lpDrawItemStruct->rcItem),
                                hBrush);
                        ::DeleteObject(hBrush);
                }
                else if (lpDrawItemStruct->itemAction & ODA_FOCUS)
                {
                        RECT focusR=lpDrawItemStruct->rcItem;
                        focusR.top+=2;
                        focusR.bottom-=2;
                        focusR.left+=2;
                        focusR.right-=2;
                        ::DrawFocusRect(lpDrawItemStruct->hDC,&focusR);
                }
        return;         // eat it
        }

        CDialog::OnDrawItem(nIDCtl, lpDrawItemStruct);
}

void CScatter3DDlg::OnKillfocusSymbolSize()
{
        if (!UpdateData())
                return;
        m_pDisplay->SetSymbolSize(m_SymbolSize);
}

void CScatter3DDlg::OnMouseRotate()
{
        UpdateData();
        m_pDisplay->AllowMouseRotate(m_bMouseRotate);
}

void CScatter3DDlg::OnAutoX()
{
        UpdateData();
        m_pDisplay->SetAutoScaleX(m_bAutoX);
        m_MaxX=m_pDisplay->GetMaxX();
        m_MinX=m_pDisplay->GetMinX();
        UpdateData(FALSE);
        ((CEdit *)GetDlgItem(IDC_MIN_X))->SetReadOnly(m_bAutoX);
        ((CEdit *)GetDlgItem(IDC_MAX_X))->SetReadOnly(m_bAutoX);
}
```

```
void CScatter3DDlg::OnAutoY()
{
        UpdateData();
        m_pDisplay->SetAutoScaleY(m_bAutoY);
        m_MaxY=m_pDisplay->GetMaxY();
        m_MinY=m_pDisplay->GetMinY();
        UpdateData(FALSE);
        ((CEdit *)GetDlgItem(IDC_MIN_Y))->SetReadOnly(m_bAutoY);
        ((CEdit *)GetDlgItem(IDC_MAX_Y))->SetReadOnly(m_bAutoY);
}
void CScatter3DDlg::OnAutoZ()
{
        UpdateData();
        m_pDisplay->SetAutoScaleZ(m_bAutoZ);
        m_MaxZ=m_pDisplay->GetMaxZ();
        m_MinZ=m_pDisplay->GetMinZ();
        UpdateData(FALSE);
        ((CEdit *)GetDlgItem(IDC_MIN_Z))->SetReadOnly(m_bAutoZ);
        ((CEdit *)GetDlgItem(IDC_MAX_Z))->SetReadOnly(m_bAutoZ);
}
void CScatter3DDlg::OnKillfocusMaxX()
{
        UpdateData();
        m_pDisplay->SetMaxX(m_MaxX);
}
void CScatter3DDlg::OnKillfocusMaxY()
{
        UpdateData();
        m_pDisplay->SetMaxY(m_MaxY);
}
void CScatter3DDlg::OnKillfocusMaxZ()
{
        UpdateData();
        m_pDisplay->SetMaxZ(m_MaxZ);
}
void CScatter3DDlg::OnKillfocusMinX()
{
        UpdateData();
        m_pDisplay->SetMinX(m_MinX);
}
void CScatter3DDlg::OnKillfocusMinY()
{
        UpdateData();
        m_pDisplay->SetMinY(m_MinY);
}
void CScatter3DDlg::OnKillfocusMinZ()
```

```
{
        UpdateData();
        m_pDisplay->SetMinZ(m_MinZ);
}
void CScatter3DDlg::OnMakeSel()
{
        m_bOldAllowRotate=m_bMouseRotate;
        m_bMouseRotate=FALSE;
        UpdateData(FALSE);
        GetDlgItem(IDC_MOUSE_ROTATE)->EnableWindow(FALSE);
        m_pDisplay->StartMakeSel();
}
void CScatter3DDlg::OnCancelSel()
{
        m_pDisplay->CancelSel();
        m_bMouseRotate=m_bOldAllowRotate;
        GetDlgItem(IDC_MOUSE_ROTATE)->EnableWindow();
        UpdateData(FALSE);
}
void CScatter3DDlg::OnZoomSel()
{
        m_pDisplay->ZoomSel();
        m_bAutoX=m_bAutoY=m_bAutoZ=FALSE;
        m_bMouseRotate=m_bOldAllowRotate;
        GetDlgItem(IDC_MOUSE_ROTATE)->EnableWindow();
        m_MaxX=m_pDisplay->GetMaxX();
        m_MinX=m_pDisplay->GetMinX();
        m_MaxY=m_pDisplay->GetMaxY();
        m_MinY=m_pDisplay->GetMinY();
        m_MaxZ=m_pDisplay->GetMaxZ();
        m_MinZ=m_pDisplay->GetMinZ();
        UpdateData(FALSE);
        ((CEdit *)GetDlgItem(IDC_MIN_X))->SetReadOnly(m_bAutoX);
        ((CEdit *)GetDlgItem(IDC_MAX_X))->SetReadOnly(m_bAutoX);
        ((CEdit *)GetDlgItem(IDC_MIN_Y))->SetReadOnly(m_bAutoY);
        ((CEdit *)GetDlgItem(IDC_MAX_Y))->SetReadOnly(m_bAutoY);
        ((CEdit *)GetDlgItem(IDC_MIN_Z))->SetReadOnly(m_bAutoZ);
        ((CEdit *)GetDlgItem(IDC_MAX_Z))->SetReadOnly(m_bAutoZ);
}
void CScatter3DDlg::OnCopy()
{
        m_pDisplay->CopyToClipboard();
}
// Opening folders to brows for input file
void CScatter3DDlg::OnLoad()
{
 // Deleting the temp. file that the programe creates
 m_combox.ResetContent();
 m_comboy.ResetContent();
 m_comboz.ResetContent();
 TRY
 {
    char *name = "C:\Documents and
Settings\default\Desktop\ward\C++\scatter3\test\data.dat";
    CFile::Remove(name);
 }
 CATCH( CFileException, e )
 {
    #ifdef _DEBUG
       afxDump << "File cannot be removed\n";
    #endif
 }
 END_CATCH
          CFileDialog dlg(TRUE,NULL,NULL);
          if (dlg.DoModal()!=IDOK)
                 return;
 FILE *fp;
 char line[STRLEN];
 int i = 0;
 CString fName = dlg.GetFileName();//"demo.dat";
 fp = fopen(fName, "r");
 file = fName;
 while (fgets(line, STRLEN, fp))
    i++;
 fseek(fp, 0, SEEK_SET);
// fcolse(fp);
 int Inputfilecolumns = 0;
 for ( i = 0; i<=1000; i++)
 if(line[i] == '\t')
 {
  Inputfilecolumns++;
 }
 CString columnelement;
 for(i = 0; i<Inputfilecolumns-1; i++)
 {
  switch( i )
  {
     case 0:
     m_combox.InsertString(i,"IP packet length");
     m_comboy.InsertString(i,"IP packet length");
     m_comboz.InsertString(i,"IP packet length");
     break;
```

```
case 1:
m_combox.InsertString(i,"IP packet traffic");
m_comboy.InsertString(i,"IP packet traffic");
m_comboz.InsertString(i,"IP packet traffic");
break;
case 2:
m_combox.InsertString(i,"IP byte traffic");
m_comboy.InsertString(i,"IP byte traffic");
m_comboz.InsertString(i,"IP byte traffic");
break;
case 3:
m_combox.InsertString(i,"IP packet rate");
m_comboy.InsertString(i,"IP packet rate");
m_comboz.InsertString(i,"IP packet rate");
break;
case 4:
m_combox.InsertString(i,"IP byte rate");
m_comboy.InsertString(i,"IP byte rate");
m_comboz.InsertString(i,"IP byte rate");
break;
case 5:
m_combox.InsertString(i,"UDP packet length");
m_comboy.InsertString(i,"UDP packet length");
m_comboz.InsertString(i,"UDP packet length");
break;
case 6:
m_combox.InsertString(i,"UDP packet traffic");
m_comboy.InsertString(i,"UDP packet traffic");
m_comboz.InsertString(i,"UDP packet traffic");
break;
case 7:
m_combox.InsertString(i,"UDP byte traffic");
m_comboy.InsertString(i,"UDP byte traffic");
m_comboz.InsertString(i,"UDP byte traffic");
break;
case 8:
m_combox.InsertString(i,"UDP packet rate");
m_comboy.InsertString(i,"UDP packet rate");
m_comboz.InsertString(i,"UDP packet rate");
break;
case 9:
m_combox.InsertString(i,"UDP byte rate");
m_comboy.InsertString(i,"UDP byte rate");
m_comboz.InsertString(i,"UDP byte rate");
break;
case 10:
```

```
          m_combox.InsertString(i,"HB End-to-end delay");
          m_comboy.InsertString(i,"HB End-to-end delay");
          m_comboz.InsertString(i,"HB End-to-end delay");
          break;
          case 11:
          m_combox.InsertString(i,"HB Packet loss rate");
          m_comboy.InsertString(i,"HB Packet loss rate");
          m_comboz.InsertString(i,"HB Packet loss rate");
          break;
     }
   }
}

void CScatter3DDlg::OnDropFiles(HDROP hDropInfo)
{
            char fName[1000];
            ::DragQueryFile(hDropInfo,0,fName,1000);
TRACE("File %s got dropped here\n",fName);

            CDialog::OnDropFiles(hDropInfo);

            DoOpen(fName);
}

#define STRLEN 1000
BOOL CScatter3DDlg::DoOpen(CString fName)
{
            CString errStr;
            int i,j,rv;
            char line[STRLEN]; //array of chars 1000 elements
            FILE *fp;
            fp = fopen(fName, "r");
            if (!fp)
            {
                        AfxMessageBox("Couldn't open input file",MB_ICONWARNING);
                        return FALSE;
            }
   //OnZoomSet()::m_para1;

            fgets(line,STRLEN,fp);
            m_DatCount=1;
  // count dimensions (number of floats in line)
            int dimCount=0;
            BOOL bInFlt=FALSE;
            for (i=0; i<int(strlen(line)); i++)
            {
```

```
            if (isspace(line[i]))
            {if (bInFlt){bInFlt=FALSE;dimCount++;}}
            else bInFlt=TRUE;
        }

        if (dimCount!=3 && dimCount!=4 && dimCount!=6)
        {
            errStr.Format("file has %d columns, but should have 3, 4 or 6",dimCount);
            AfxMessageBox(errStr,MB_ICONWARNING);
            fclose(fp);
            return FALSE;
        }

// find number of lines
        while (fgets(line, STRLEN, fp))
            m_DatCount++;

        if (m_pData!=NULL)
            delete [] m_pData;
        if (m_pColList!=NULL)
        {
            delete [] m_pColList;
            m_pColList=NULL;
        }

        m_pData=new float[m_DatCount*3];
        if (dimCount!=3)
            m_pColList=new COLORREF[m_DatCount];

// rewind file
        fseek(fp, 0, SEEK_SET);

        // load data
        int readCount=0;
        int col[3];
        for (i=0; i<m_DatCount; i++)
        {
            for (j=0; j<3; j++)
            {
                if ((rv=fscanf(fp, "%f", &(m_pData[i*3+j])))!=1)
                    goto error;
            }
            if (dimCount==4)        // COLORREF as last item
            {
                if ((rv=fscanf(fp, "%u", &(m_pColList[i])))!=1)
                    goto error;
        }
        else if (dimCount==6)  // r,g,b given separately
        {
            for (j=0; j<3; j++)
            {
                if ((rv=fscanf(fp, "%u", &(col[j])))!=1)
                    goto error;
            }
            m_pColList[i]=RGB(col[0],col[1],col[2]);
        }
        else
        {
            ASSERT(dimCount==3);
            ;
        }
    }
    fclose(fp);
    m_pDisplay->SetData(m_DatCount,RGB(255,0,0),m_pData,m_pColList);
    UpdateData(FALSE); // update count

    if (dimCount==3)
    {
        GetDlgItem(IDC_PT_COL_TXT)->EnableWindow();
        GetDlgItem(IDC_PT_COLOUR)->ShowWindow(SW_SHOWNA);
    }
    else
    {
        GetDlgItem(IDC_PT_COL_TXT)->EnableWindow(FALSE);
        GetDlgItem(IDC_PT_COLOUR)->ShowWindow(SW_HIDE);
    }

    return TRUE;

error:
    AfxMessageBox("Error reading data file");
    fclose(fp);
    return FALSE;
}
// changing the point color
void CScatter3DDlg::OnPtColour()
{
    CColorDialog dlg;
    if (dlg.DoModal()!=IDOK)
        return;

    m_PtCol=dlg.GetColor();      // for drawing button
```

```
        GetDlgItem(IDC_PT_COLOUR)->Invalidate();
        m_pDisplay->SetPtCol(m_PtCol);
}
void CScatter3DDlg::OnButton1()
{
        // TODO: Add your control notification handler code here
   FILE *input;
   FILE *output;
   float coordaxis[12];
   int pointcolor = 0;
   // initializing the carrier array
   for(int i = 0; i <=12 ; i++)
      coordaxis[i] = 0.0;
   CString in, out;
   out = "data.dat";
   input = fopen(file,"r");  // file - is a global variable gets assigned in onload method
   output = fopen(out,"w");

   int xx = m_combox.GetCurSel();     int yy = m_comboy.GetCurSel();
   int zz = m_comboz.GetCurSel();
   // start reading from the assigned file in the OnLoad
fscanf (input,"%f%f%f%f%f%f%f%f%f%f%f%f%d", &coordaxis[0], &coordaxis[1],
&coordaxis[2], &coordaxis[3], &coordaxis[4], &coordaxis[5],  &coordaxis[6],
&coordaxis[7], &coordaxis[8],  &coordaxis[9], &coordaxis[10], &coordaxis[11],
&pointcolor
        );

   while(!feof(input))
   {
      // Output only the fields in the selected combo boxs
      fprintf( output, "%f\t%f\t%f\t%d\n",      coordaxis[xx] , coordaxis[yy] , coordaxis[zz] ,
pointcolor  ) ;

     // read the next line up in the data file
    fscanf (input,"%f%f%f%f%f%f%f%f%f%f%f%f%d", &coordaxis[0], &coordaxis[1],
&coordaxis[2],  &coordaxis[3], &coordaxis[4], &coordaxis[5], &coordaxis[6],
&coordaxis[7], &coordaxis[8],  &coordaxis[9], &coordaxis[10], &coordaxis[11],
&pointcolor
           );
   }
   fclose(input);
   fclose(output);
   //m_combox
   DoOpen("data.dat");
}
```

# REFERENCES

[1] Anderson, D., Frivold, T. & Valdes, A (May, 1995). Next-generation Intrusion Detection Expert System (NIDES): A Summary. SRI International Technical Report SRI-CSL-95-07.

[2] Carpenter, G.A. & Grossberg, S. (1987). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. Computer Vision, Graphics and Image Processing 37, 54-115.

[3] Chung, M., Puketza, N., Olsson, R.A., & Mukherjee, B. (1995) Simulating Concurrent Intrusions for Testing Intrusion Detection Systems:Parallelizing. In NISSC. pp. 173-183.

[4] Cramer, M., et. al. (1995). New Methods of Intrusion Detection using Control-Loop Measurement. In Proceedings of the Technology in Information Security Conference (TISC) '95. pp. 1-10.

[5] Debar, H., Becke, M., & Siboni, D. (1992). A Neural Network Component for an Intrusion Detection System. In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy.

[6] Debar, H. & Dorizzi, B. (1992). An Application of a Recurrent Network to an Intrusion Detection System. In Proceedings of the International Joint Conference on Neural Networks. pp. (II)478-483.

[7] Denault, M., Gritzalis, D., Karagiannis, D., and Spirakis, P. (1994). Intrusion Detection: Approach and Performance Issues of the SECURENET System. In Computers and Security Vol. 13, No. 6, pp. 495-507

[8] Denning, Dorothy. (February, 1987). An Intrusion-Detection Model. IEEE Transactions on Software Engineering, Vol. SE-13, No. 2.

[9] Fox, Kevin L., Henning, Rhonda R., and Reed, Jonathan H. (1990). A Neural Network Approach Towards Intrusion Detection. In Proceedings of the 13th National Computer Security Conference.

[10] Frank, Jeremy. (1994). Artificial Intelligence and Intrusion Detection: Current and Future Directions. In Proceedings of the 17th National Computer Security Conference.

[11] Fu, L. (1992). A Neural Network Model for Learning Rule-Based Systems. In Proceedings of the International Joint Conference on Neural Networks. pp. (I) 343-348.

[12] Hammerstrom, Dan. (June, 1993). Neural Networks At Work. IEEE Spectrum. pp. 26- 53.

[13] Helman, P., Liepins, G., and Richards, W. (1992). Foundations of Intrusion Detection. In Proceedings of the Fifth Computer Security Foundations Workshop pp. 114-120.

[14] Helman, P. and Liepins, G., (1993). Statistical foundations of audit trail analysis for the detection of computer misuse, IEEE Trans. on Software Engineering, 19(9):886-901.

[15] Ilgun, K. (1993). USTAT: A Real-time Intrusion Detection System for UNIX. In Proceedings of the IEEE Symposium on Research in Security and Privacy. pp. 16-28.

[16] Kohonen, T. (1995) Self-Organizing Maps. Berlin: Springer.

[17] Kumar, S. & Spafford, E. (1994) A Pattern Matching Model for Misuse Intrusion Detection. In Proceedings of the 17th National Computer Security Conference, pages 11-21.

[18] Kumar, S. & Spafford, E. (1995) A Software Architecture to Support Misuse Intrusion Detection. Department of Computer Sciences, Purdue University; CSD-TR-95-009

[19] Lunt, T.F. (1989). Real-Time Intrusion Detection. Computer Security Journal Vol. VI, Number 1. pp. 9-14.

[20] Mukherjee, B., Heberlein, L.T., Levitt, K.N. (May/June, 1994). Network Intrusion Detection. IEEE Network. pp. 28-42.

[21] Porras, P. & Neumann, P. (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In Proceedings of the 20th NISSC.

[22] Puketza, N., Chung, M., Olsson, R.A. & Mukherjee, B. (September/October, 1997). A Software Platform for Testing Intrusion Detection Systems. IEEE Software, Vol. 14, No. 5

[23] Ryan, J., Lin, M., and Miikkulainen, R. (1997). Intrusion Detection with Neural Networks. AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop (Providence, Rhode Island), pp. 72-79. Menlo Park, CA: AAAI.

[24] Sebring, M., Shellhouse, E., Hanna, M. & Whitehurst, R. (1988) Expert Systems in Intrusion Detection: A Case Study. In Proceedings of the 11th National Computer Security Conference.

[25] Staniford-Chen, S. (1995, May 7). Using Thumbprints to Trace Intruders. UC Davis.

[26] Tan, K. (1995). The Application of Neural Networks to UNIX Computer Security. In Proceedings of the IEEE International Conference on Neural Networks, Vol.1 pp. 476-481.

[27] Tan, K.M.C & Collie, B.S. (1997). Detection and Classification of TCP/IP Network Services. In Proceedings of the Computer Security Applications Conference. pp. 99-107.

[28] White, G.B., Fisch, E.A., and Pooch, U.W. (January/February 1996). Cooperating Security Managers : A Peer-Based Intrusion Detection System. IEEE Network. pp. 20-23.

[29] ANS. ARTs: ANSnet Router Statistics Software, 1992

[30] J. Fleiss. Statistical Methods for Rates and Proportion. John Wiley & Sons, 1981.

[31] W. Cochran. Sampling Techniques. John Wiley & Sons, 1987.

[32] Grimes, Stockton, Reilly and Templeman. Beginning ATL COM Programming. Wrox Press, 1998.

[33] Michael J. Young. Mastering Visual C++ 6. SYBEX Inc, 1998.

[34] Johannes Gehrke. Database Management Systems. McGraw-Hill Higher Education, 2000.

[35] ComponentOne LLC. True DBGrid Pro 7.0. ComponentOne ™ LLC, 2000.

[36] David J. Kruglinski, George Shepherd. Programming Microsoft Visual C++ Fifth Edition. Microsoft Press, 1998.

[37] Microsoft Corporation. Visual C++ 6.0 MFC Library Reference, Microsoft Press, 998.