On Network Latency in Distributed Interactive Applications

J. D. Delaney, Department of Computer Science
T. Ward and S. McLoone, Department of Electronic Engineering

This paper has three objectives. Firstly it describes the historical development of Distributed Interactive Applications. It then defines network latency. Finally it describes a new approach to masking network latency in Distributed Interactive Applications called the strategy model approach. This approach derives from the on-going PhD studies of one of the authors. A software application to gather strategy data from users is described in detail and an example of deriving a user strategy is given.

***Key Terms**: Distributed Interactive Applications, Latency, Strategy Model*

1. Introduction

Human interaction with computers has evolved from batch processing using Hollerith cards to virtual reality systems in which the user can be fully immersed. The origins of virtual reality technology can be traced back to vehicle simulation research in the 1920s[i] and since then, virtual environments have spread to areas such as military simulations, cooperative whiteboards and architectural design. The first distributed software virtual environment was SIMNET, a United States research program initiated in 1983 to train soldiers in battlefield tactics[ii]. Since then numerous distributed virtual worlds and applications have been documented[iii, iv]. We refer to these systems as Distributed Interactive Applications (DIAs). They are networked software environments that usually model a real environment to facilitate communicative, cooperative and collaborative tasks involving numerous simultaneous participants[v].

This paper provides a historical context to the development of DIAs, focusing on the development of standards within the military domain and on the development of research virtual environments in academia. It then describes the network latency phenomenon and presents a new approach to masking latency called strategy modeling. A novel software tool for collecting actual data from users is described and an example is given of how this data can be analyzed to derive a user strategy. This work is based on one of the authors PhD research.

The paper is structured as follows: the following section provides the historical background. Latency is defined in section 3. In section 4 the strategy approach to masking latency is presented and the software tool for collecting user strategy data is described in section 5. Section 6 contains an illustrative example. The paper ends with a concluding discussion.

2. Historical Background

In this section we will give a succinct summary of the development of DIAs within the military and academic domains. Distributed applications have their origins in United States military research in the early 1980s and since the early 1990s many academic institutions have developed research groups, often with the participation of industry or government agencies. Running in parallel with both the military and academic research has been research into networked games, beginning with a program called *Flight* in 1984. Interested readers are referred to Singhal and Zyda[vi], McCoy[vii] and Smed[viii].

From Military Research to International Standards

Research into military DIAs has been almost exclusively American-based[ix]. The US department of defense refers to military DIAs as Advanced Distributed Simulations (ADS)[x and xi]. ADS aims to support three different types of simulation:

- *live*: involving people interfacing with real equipment. This equipment (hardware) is itself linked into the simulation environment. Such live participants are also referred to as Man-In-The-Loop (MITL) or Hardware-In-The-Loop (HITL);
- *virtual*: involving people interacting with simulated equipment. The simulated equipment is often referred to as a *simulator*;
- *constructive*: entirely computer generated simulation i.e. simulated people operating in a simulated environment. The term Semi-Automated Forces (SAF) is frequently used to describe this type of simulation.

The evolution of ADS can be described in terms of the evolution of simulation architectures that define the standards and protocols for interoperability between participating simulations. We will look at the evolution of these architectures as they are illustrated in Figure 1.

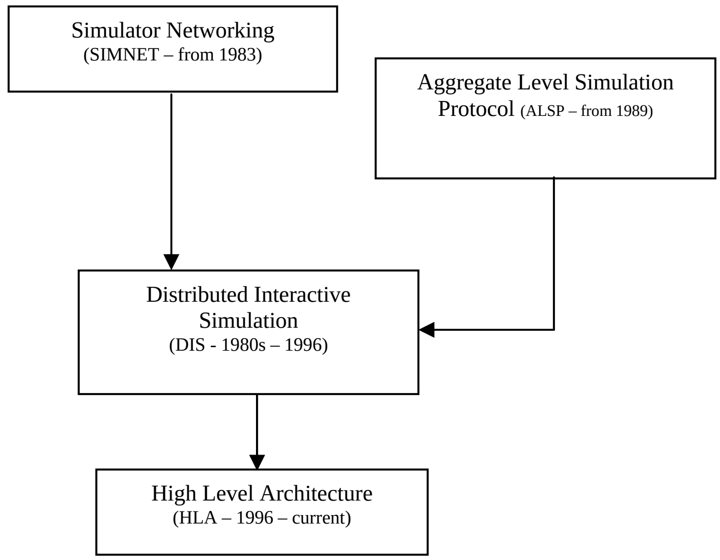


Figure 1: Evolution of architectures

SIMNET

SIMulation NETwork (SIMNET) was a US Defense Advanced Research Projects Agency (DARPA) funded programme, which ran from 1983 until 1990 and resulted in the SIMNET protocol, an informal protocol developed by Bolt, Beranek and Newman (BBN). The SIMNET project aimed to provide a simulation environment suitable for tactical training of tank crews. The significant contribution of the SIMNET programme was to raise the importance of team training, and the use of simulation technology to support such training. The resulting network of tank simulators facilitated regular and intensive practice in combat skills. In addition, this also provided an environment capable of evaluating new and emerging tactics, doctrines and weapons systems. Many of the eventual 250 SIMNET simulators remain in operation with the US Army.

ALSP

Interoperability issues with other simulators were not a key concern for SIMNET, since the SIMNET simulators were specially developed and used a common set of protocols to exchange information. This was not, however, the case in other areas where the US Department of Defence (DoD) wanted to use such technology. In 1990 DARPA sponsored MITRE[xii] to investigate the design of a general protocol to support interoperability of *existing* aggregate-level constructive combat simulations. The term *aggregate* refers to the ability of a simulation to represent groups of entities (e.g. squadron) while preserving the capabilities of the composite entities making up the aggregate. The resulting protocol became known as Aggregate Level Simulation Protocol (ALSP). The ALSP project recognized that different branches within the military had already invested in development of their own combat simulations. Increasingly, given the experiences of SIMNET, it was seen as beneficial to allow these disparate simulations to interoperate in ways never intended - ideally without having to totally redevelop them.

DIS

Features of both SIMNET and ALSP projects contributed to the subsequent development of the Distributed Interactive Simulation (DIS) protocol as IEEE std. 1278.1-1995. The DIS protocol is a "government/industry initiative to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual worlds for the simulation of highly interactive activities."[xiii] The DIS standard defines the standard data messages that are exchanged between simulation applications. It only supports messages for certain domains, although it can be extended to include new domains. Sample domains are radio communications, warfare and entity information/interaction. Data messages are known as Protocol Data Units (PDU) and this is an application layer protocol. The most important PDU in the DIS standard is the Entity State Protocol Data Unit (ESPDU). This contains information regarding the entity state (type, position, velocity, visual appearance, capabilities etc) and accounts for 96% of DIS network traffic. The architecture is fully replicated with no central control of the simulation exercise. This means that without a data reduction scheme, a simulation involving 100,000 players would require 375Mbits/s of network bandwidth to each participating node[xiv]. The DIS standard therefore defines the Dead Reckoning concept for reducing network traffic. Despite this the DIS protocol suffers from scalability issues.

HLA

In 1991 the US DoD formed the DMSO (Defense, Modeling and Simulation Office) to consolidate the efforts in ADS development. The cornerstone of DMSO efforts is the High Level Architecture (HLA)[xv]. The HLA learned lessons from its predecessors and moved away from defining the format of data interchange. The focus moved from specifying communication content (the DIS approach) to the communication mechanism. In 1998 the HLA was adopted as an international standard by the Object Management Group (OMG), and was subsequently accepted and defined in IEEE std. 1516-2000. It is based on the premise that no one simulation can satisfy all uses and users. In HLA terms an individual simulation is referred to as a *federate*. A group of federates that intend to interoperate with one another form a *federation*. This terminology was introduced to avoid excessive use of the term *simulation*, where it would be unclear whether *a simulation* refers to the individual or the whole.

The HLA has two stated goals:

(1) to achieve interoperability of a broader range of types of simulations than previous ADS achieved;

(2) to maximize the reuse of existing code and architecture framework.

The HLA is defined by three components: (a) an Interface Specification to define federate interoperability, (b) a set of rules to govern federate interaction and (c) an *Object Model Template*.

SISO

As with its DIS predecessor, the HLA was initially intended for use within the US military. There was, however, increasing awareness that the HLA was applicable to problems in the wider simulation community, both outside of the US and outside of the defence community. Since 1996 the Simulation Interoperability Standards Organization (SISO)[xvi] has been at the fore in reporting ADS developments. SISO provides a focus for all groups with an interest in ADS developments. The SISO currently has 1176 official members representing over 400 organizations covering broad representation (approximately 51% commercial, 41% government/military and 8% academia). The members represent 12 countries including Canada, France, Germany, Ghana, Israel, Japan, Netherlands, Spain, Sweden, United Kingdom and the USA. A good summary of standards in collaborative virtual environments is given by Pullen et al[xvii].

Academic Distributed Interactive Applications

Academic research into DIAs originated with the NPSNET in 1990. To review all subsequent academic DIAs would be a daunting task. Instead we will describe the origins and development of the NPSNET, and then focus on a number of more recent DIAs. Some important academic DIAs upon which all subsequent DIAs are constructed include MASSIVE[xviii and xix] (University of Nottingham), PaRADISE[xx] (Stanford University), DIVE[xxi] (Swedish Institute of Computer Science) and SPLINE[xxii] (Mitsubishi Electric Research Laboratories). These have been extensively documented elsewhere and interested readers are referred to the relevant references.

NPSNET

The origins of this system date back to a missile virtual environment called FOG-M developed at the Navy Postgraduate School (NPS) in 1986. This was networked with a target vehicle simulator called VEH and developed into the Moving Platform Simulator (MPS). This then grew into NPSNET-1 in 1990, leading to NPSNET-IV[xxiii] in 1993 and subsequently developed into NPSNET-V component framework[xxiv] in 2000. NPSNET-V is a platform for student development and a test bed for advanced virtual environment research. It is a Java[xxv] component-based architecture that is extensible, allows dynamic extensibility (runtime replacement of any part of the application), is scalable, provides for cross-version compatibility and allows persistent storage. It has built upon aspects of other architectures such as Bamboo, CORBA, GNU/MAVERIK, Deva, JADE (Java Adaptive Dynamic Environment), ERCSP (Extensible Runtime Containment and Services Protocol) and JavaBeans. NPSNET-V allows the construction of applications as component hierarchies rooted in an invariant microkernel. Components communicate with each other through an interface layer and event model.

ATLAS

This Java/C++-based scalable network framework for DIAs was developed at the Information and Communications University in Korea[xxvi]. To improve scalability the authors identify four core issues: (1) communication architecture – peer-to-peer vs. peer-to-server model, (2) Interest Management – relevance filtering and area-of-interest management[xxvii] (3) Concurrency Control – replication means that replicas must be synchronized to avoid an inconsistent virtual world and (4) Data Replication – a central server maintains the actual virtual world and clients replicate all or part of this world[xxviii]. ATLAS has two main components – a Server and a Peer component. The Server Component consists of four managers: session, region, event and communication managers. Likewise the Peer consists of the same four managers together with the client application. ATLAS provides no graphics or UI support but it has a *veneer layer* that allows it to integrate successfully with other virtual environments. It has been integrated with systems such as the single user virtual reality system Kitten for a collaborative engineering application and Virtual Playground.

MOVE

This is a 3D collaborative environment built upon a component groupware framework called ANTS, which provides collaborative services. It is being developed by the universities of Rovira i Virgili and Murcia in Spain as part of the Catalonian Internet2 project[xxix]. Users can interact with other users or with other entities such as documents, whiteboards or voting tools. MOVE has been developed using open technologies such as Java, the Virtual Reality Modeling Language (VRML)[xxx], and the H-Anim avatar standard specification[xxxi]. MOVE aims to be scalable, component-based and record all avatar behaviour while in the virtual world. To tackle these problems it uses the standard JavaBeans specification as the component model and defines an event monitoring/handling system to facilitate the collection and retrieval of behavioral data.

SCORE

This DIA was developed at the L'Universite de Nice-Sophia Antipolis, France. The virtual world is divided into cells, each of which has a multicast group. Each user avatar has a square area of interest and subscribes to the multicast groups that its area of interest intersects. The cell size can be static or dynamic. The dynamic cell size can be calculated based on parameters such as avatar spatial density or available multicast addresses. The cells may also be of different size, allowing a fine grid where avatar spatial density is high[xxxii].

Having reviewed the historical foundations of DIAs, the next section will describe an inherent network phenomenon called latency which is the fundamental technical limitation associated with all DIAs.

3. Network Latency

There are several potential delays in a distributed system, including tracking delay, computation delay, graphics rendering delay, synchronization delay, communication delay and display delay[xxxiii]. One of the limiting factors to deploying interactive applications with possibly unrestrained geographical distribution is the communication delay associated with the interconnecting network. Information takes a finite, unpredictable length of time to reach each user because of the heterogeneous nature of the intervening networks. This inherent time delay is called *latency* or lag. Latency is the time it takes for a packet of information to travel from an application on one computer to an application on another computer across a network. Munki[xxxiv] provides a formula for calculating latency as follows:

*Latency = time per bit * number of bits + overhead (routing etc)*

Some typical latency values for different network technologies are shown in Table 1.

Network	Travel time	Time per data unit
Ethernet LAN	0-3 ms	Insignificant
Modem to modem	~ 100 ms	~ 1 ms/byte
Analogue modems	~ 200 ms	~ 2ms/byte
ISDN to ISDN	~ 10 ms	~ 0.5ms/byte
Internet	up to seconds	highly variable

Table 1: Some typical latency values

Acceptable latency values for DIS applications are defined by the Communications Architecture for Distributed Interactive Simulation (CADIS). These are illustrated in Figure 2 for an ISO protocol stack model[xxxv]. The latency experienced by the end user is the application-to-application delay.

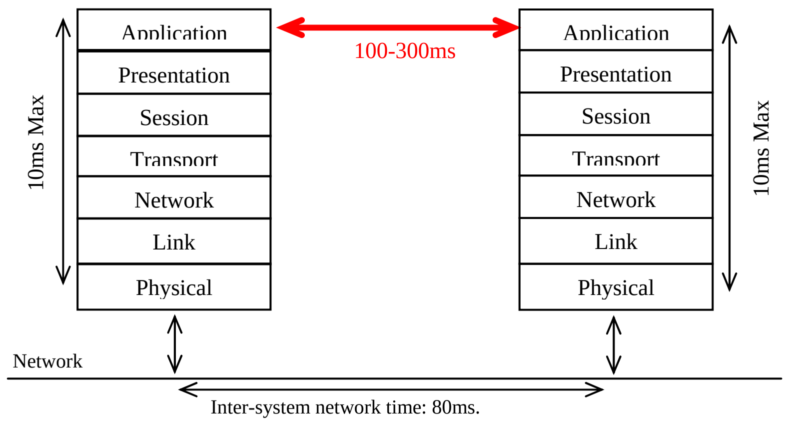


Figure 2: ISO Protocol Stack and CADIS latency values

Latency is an inherent network characteristic. The information travelling between two networked computers connected via fibre optic cable travels at about two-thirds the speed of light in a vacuum. Based on this, Cheshire[xxxvi] gives some minimum theoretical latency values as a function of distance. A rule of thumb is that round-trip latency is 1ms per 100km. For example, a minimum round-trip latency of 220ms is experienced in communicating with someone on the far side of the planet (a distance of about 22,000km).

Latency is particularly troublesome in interactive applications where the network delays are comparable to the interaction time[xxxvii]. Typical latency values to allow real-time distributed collaboration fluctuate between 40 and 300ms[xxxviii]. A good benchmark value is a one-way latency value of 100ms.

Latency can be reduced by physically increasing the available bandwidth. Alternatively, making more efficient use of existing bandwidth can reduce latency. Much research to date has focused on latency masking algorithms that reduce the number of network packets transmitted, creating the illusion of interactivity and ensuring a consistent worldview for all participants. Examples include client-prediction contracts (e.g. dead reckoning[xxxix]), relevance filtering[xl], packet bundling/data compression[xli], multi-resolution techniques[xlii] and time management[xliii and xliv]. There are other ways of reducing latency effects including choice of transmission protocol[xlv], network architecture[xlvi and xlvii] and Quality of Service (QoS)[xlviii and xlix].

In the next section we describe a novel client-prediction approach that the authors have developed to reduce the number of packets being transmitted between participants and in so doing mask network latency.

4. Strategy Model Approach

A new and innovative approach to masking network latency is being developed by the authors. The following paragraphs describe the work to date. For the convenience of the reader we first define some necessary terminology.

Terminology

A *goal* is the aim or objective a person has in moving through an environment. For example, the goal might be to go from point A to point B. In achieving a goal a person can adopt a number of strategies, so that any one *strategy* is an expression of the goal. Strategies can be either *steady-state* or *transient*. A steady-state strategy is a user-preferred method for achieving a set goal. A transient strategy is an exploratory method in an attempt to reach a steady-state strategy.

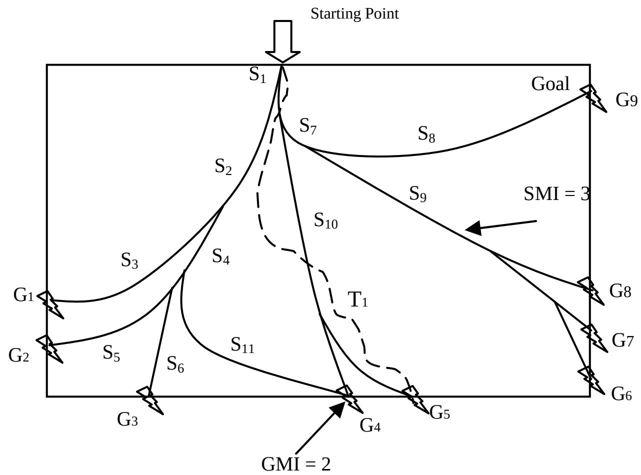


Figure 3: Terminology – S1 to S11 are strategies;G1 to G9 are goals; T1 is a sample trajectory; GMI is the Goal Multiplicity Index – how many strategies reach that goal; SMI = Strategy Multiplicity Index – how many goals this strategy lead to.

The idea underlying the hunt for strategies is to train a system to expect certain strategies based on past user behavior or based on expected user behavior. Each strategy comprises one or more trajectories – a set of trajectories can be associated with any strategy. A *trajectory* is a particular expression of a strategy. Multiplicity can refer to either strategies or goals. *Strategy Multiplicity Index* (SMI) refers to the number of goals a strategy leads to. *Goal Multiplicity Index* (GMI) refers to the number of steady-state strategies that lead to the goal. This terminology is illustrated in Figure3. In this paper we present results for a GMI of 1.

The Strategy Model Explained

A user in a virtual environment constructs a mental model of the 'physical' structure of the environment as a function of the time spent in the environment. To achieve a goal within the environment the user adopts one of possibly many potential strategies in attempting to reach the desired goal. For example a user will enter a room with the goal of walking to the nearest window. How the user gets there (the strategy adopted) will depend on furniture in the room, whether the user can see the window from the starting position and so on.

The strategy model approach relies on the creation of a library of possible strategies based on the structure of the virtual environment or on previous user behavior. The behavior of each user is observed to determine which of the existing library strategies best fits the observations. All users of the DIA share the same strategy library. As a result, users only need to communicate their adopted strategy to other users across the network. This results in a decrease in network traffic, as users do not have to update their positions continuously. Instead they only send an update packet when they change strategy. The strategy approach is used in conjunction with the dead reckoning approach as explained by Delaney et al[1].

The strategy model approach relies on being able to identify user strategies and goals. There are a number of issues to be dealt with:

- how to determine the possible steady-state strategies given the environment. If there is only one possible strategy, the problem is relatively straightforward. If there are many potential strategies, the problem then becomes that of parameterising each strategy and choosing the statistically most probable one;
- once a strategy has been chosen the length of time over which the strategy is valid must be estimated;
- the user's goal must be determined.

The first step in researching the validity of this new approach is to establish a strategy library based on the environment of the virtual world. In the next section the authors describe a software tool that allows them to record data from users navigating under controlled conditions in a virtual environment to illustrate the principle of the strategy approach.. From this, the possible steady-state strategies can be determined and a strategy library constructed.

5. Strategy Software Tool

The strategy software tool is a game-like Java application that allows the construction of a virtual environment within which users can navigate. All user actions are recorded for analysis. The software tool is called 'Universe Mapper' and will be described in the following paragraphs.

THE GRAPHICAL USER INTERFACE (GUI)

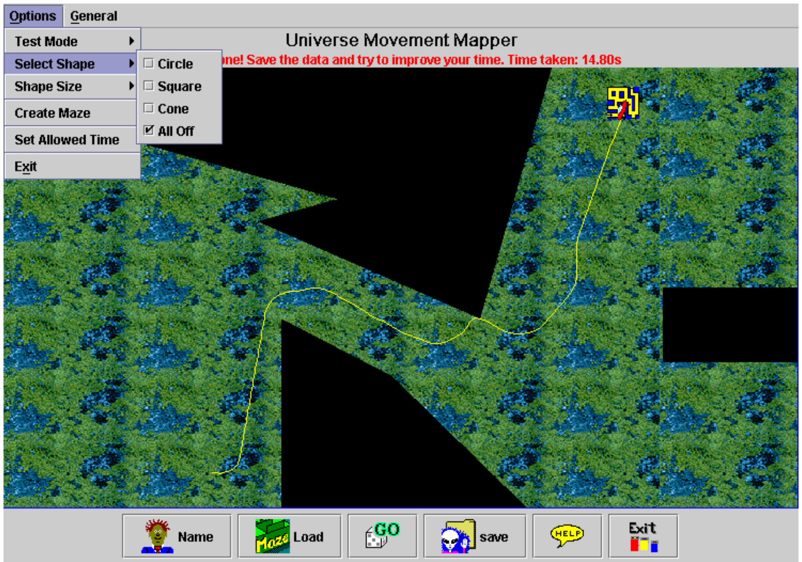


Figure 4: The Graphical User Interface of the program showing obstacles (in black), the drop down menu options and a sample user trajectory with the goal icon.

The software GUI is illustrated in Figure 4. The main options are illustrated as icons at the bottom of the screen and include Name (to input a user name), Load (to load a new environment) Go (to reset the program to an initial state), Save (to save all user movement data), Help (Hypertext document) and Exit. Users navigate a spacecraft using the arrow keys and their objective is to reach and move into the icon representing the goal. Areas presented in black are no-go areas through which users cannot pass. The spacecraft leaves a limited yellow trail in its wake so that users know where they have been over the recent past.

Some other options are included as menu items and allow the environmental parameters to be varied as well as allowing a new maze to be created. These will be explained further in the following paragraph.

THE SOFTWARE FEATURES

The software has a number of functional features that are not apparent to the users of the system. These are listed and described in Table 2.

Functional Feature	Description
Maze Construction	A Maze can be constructed using the mouse and saved to file. Start and end point locations can be set. This allows the construction of different mazes with different goals and different strategies. This feature is illustrated in Figure 5.
Limited Viewing Area	The user view of the maze environment is restricted to a region surrounding their current position. This region can be circular (as shown in Figure 6), square or conical. The size of this area can also be adjusted.
Best Scores List	To maintain user interest a best scores list based on the time to reach the goal is saved to persistent storage. This acts as an incentive to optimize the strategy adopted by users in reaching the goal.
Test Mode	A test mode is provided which allows users the opportunity to experience the dynamics of the spacecraft.
Time Allowed	The maximum time allowed by a user to reach the goal can be set manually. This is necessary because different environments with different goals will necessitate different strategies and hence different times to reach the goal. When the maximum time is exceeded the application halts and data must be saved to file.

Table 2: Universe Mapper Functional Features

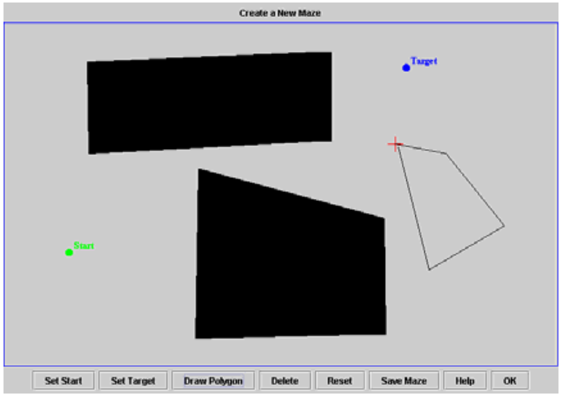


Figure 5: The 'Create Maze' option. A maze can be constructed, the start and goal points set and the information saved to file.

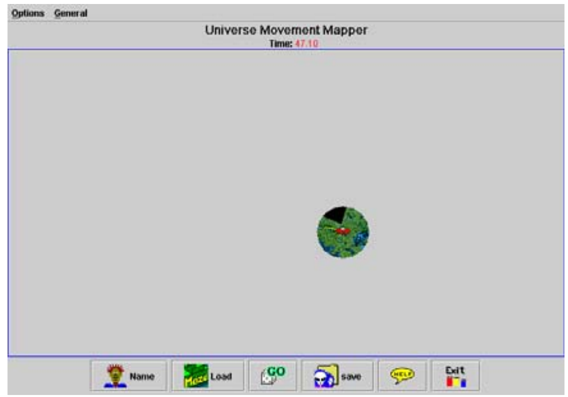


Figure 6: The user entity and the limited visible area surrounding the entity

THE SOFTWARE ARCHITECTURE

The software was written using the Java programming language. Java was chosen so that the application can be executed over the World Wide Web as an *applet*. The code structure is object-oriented and the classes, together with their principle relationships and methods, are shown in the UML diagram, Figure 7. The Frame classes (UniverseFrame and CreateMazeFrame) extend the JFrame class and the Panel classes (UniversePanel and CreateMazePanel) extend the JPanel class. All drawings are performed in the panels and the panel is contained within the frames. The frames control user input and menus.

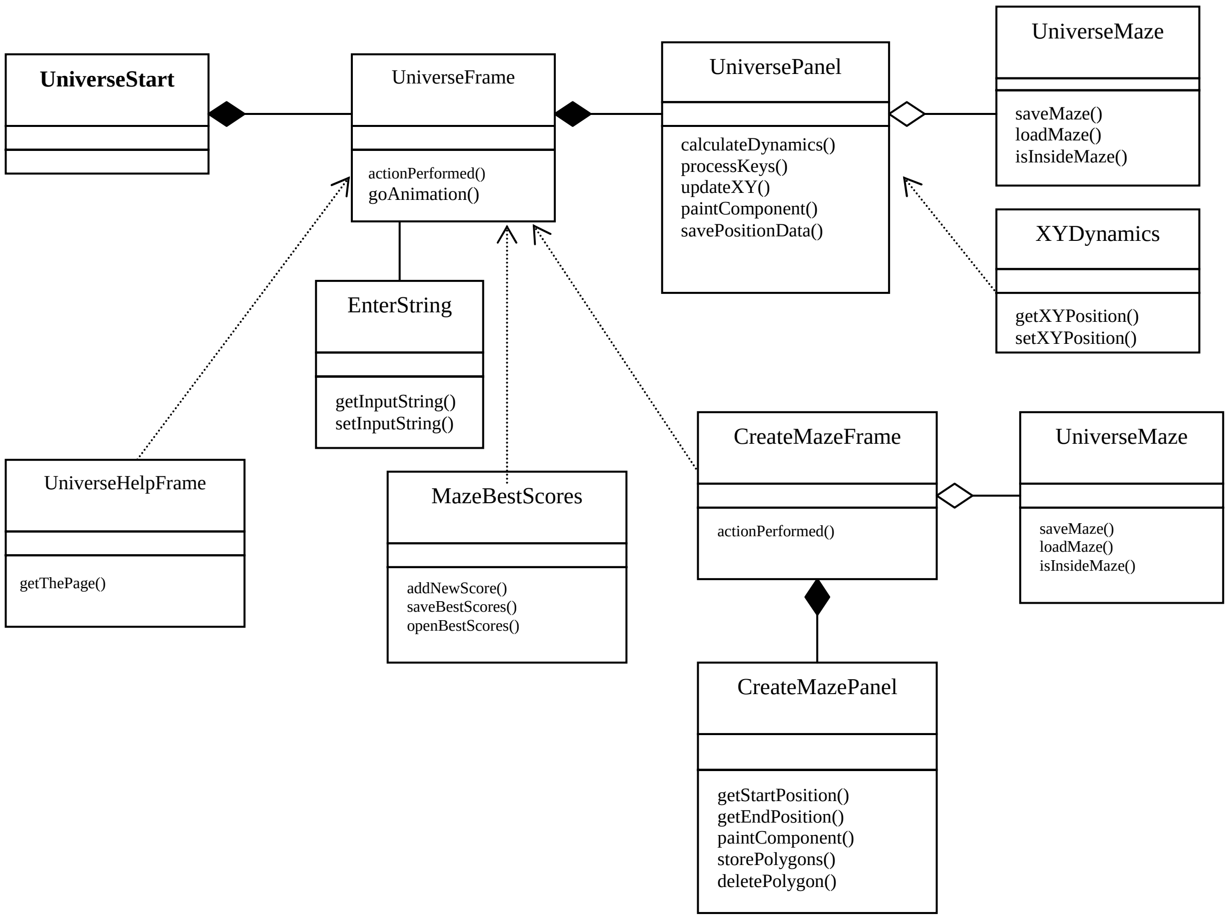


Figure 7: UML class diagram for the Universe Mapper

6. An Illustrative Example

User strategies within an environment were constructed using real data collected by means of the Universe Mapper software application. The example presented is based on the scenario of a single steady-state strategy and a single goal. The maze was designed so that the user had only one possible steady-state strategy to achieve the goal. This also means that both the steady-state user strategy and the user goal are known apriori.

Data Recording

The software tool records user trajectories based on an experimental protocol described in Appendix A. Trajectories were recorded from 15 participants. The number of trajectory data files recorded for each participant varied because they converged to a steady-state strategy after fewer attempts. This can be expressed as follows:

$$\left| T^{t_n} - T^{t_{n-1}} \right| \leq \delta$$

where

t_n is the nth trajectory associated with a strategy;
T is the time taken to reach the goal;
n is the trajectory index for a single strategy;
δ is an acceptable scalar error tolerance value.

All trajectories were analyzed using Matlab[li] to determine both transient and steady-state strategies. The maze chosen for the experiment is shown in Figure 4 and it suggests only one obvious steady-state strategy. The goal is static and consisted of reaching a particular location in the shortest time possible. Given such a goal, it is to be expected that after an initial *transient strategy*, the user will settle down to a *steady-state strategy*.

DATA ANALYSIS

As a preliminary analysis the time taken to achieve the goal as a function of the number of attempts taken was plotted. This is illustrated in Figure 8. In general the time taken to achieve the goal decreased with the number of attempts. One user failed to improve and was very erratic, achieving the goal on one occasion and then being unable to decrease the time any further. These plots support the current hypothesis regarding novel user behavior in virtual environments[lii].

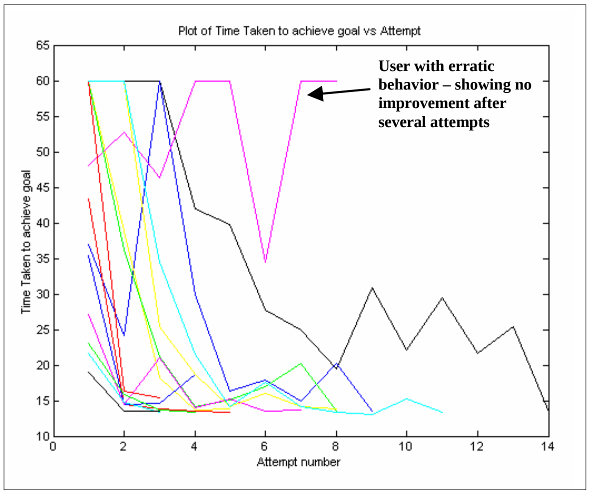


Figure 8: Plot of Time Taken to achieve goal as a function of the number of attempts made

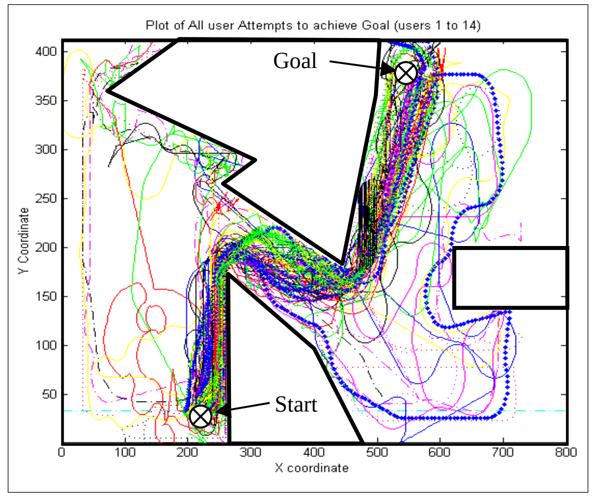


Figure 9: Spatial plot of all user trajectories recorded

A second plot (Figure 9) consists of the trajectories recorded for all the participants. The obstacles representing the maze are superimposed on the plot. From this plot it is clear that certain areas are traversed more frequently than others. In order to understand this more clearly the plot was broken into two parts – (1) Figure 10 shows the transient trajectories (those that failed to reached the goal in the allowed time) and (2) Figure 11 shows the steady-state trajectories (those that reached the goal in the time allowed).

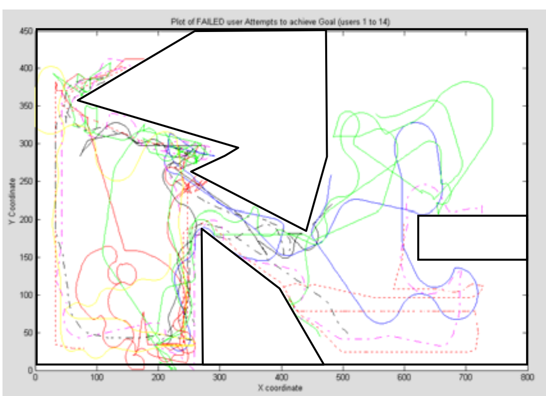


Figure 10: Spatial plot of all user trajectories that failed to reach the goal

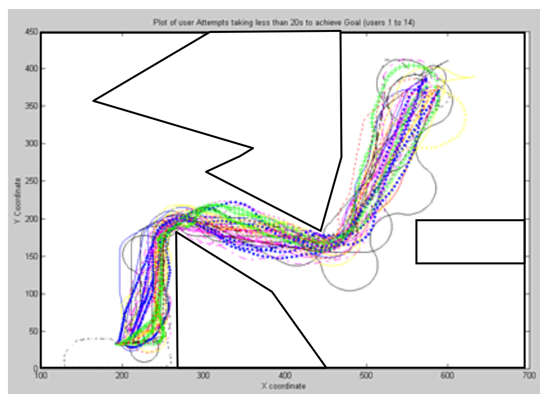


Figure 11: Plot of trajectories that reached goal via a steady-state strategy

From observation of Figures 9 to 11 we note the following:

1. there is a predominant strategy adopted to achieve the goal. The steady-state trajectories converge to a single recognizable steady-state strategy;
2. transient trajectories are not totally random and transient strategies can be identified. For example there is evidence of sweeping back and forth / up and down over the maze area; some users follow the maze boundaries to maintain a static reference and become familiar with the environment; users almost always turn into gaps instead of continuing in a straight line; open space are avoided;
3. when the goal was achieved once, the trend of the steady-state trajectories was the same in all cases.

The steady-state strategy can be derived from the steady-state trajectories shown in Figure 11. For illustrative purposes we derive a model of the steady-state strategy using polynomials of degrees 3, 5 and 7. Other models will be explored in future work. The steady-state trajectories together with the fitted polynomials are shown in Figure 12. The arithmetic average of the trajectories is also shown. The coefficients of the polynomials are listed in Table 3.

Polynomial Degree	Coefficients of Polynomials
3	0.0000, -0.0224, 8.7912, -976.3743
5	-0.0000, 0.0000, -0.0006, 0.1781, -18.1966, 260.
7	-0.0000, 0.0000, -0.0000, 0.0000, -0.0000, 0.0006, -0.0774, 3.8862

Table 3: Parameters for various polynomial fits to trajectory data

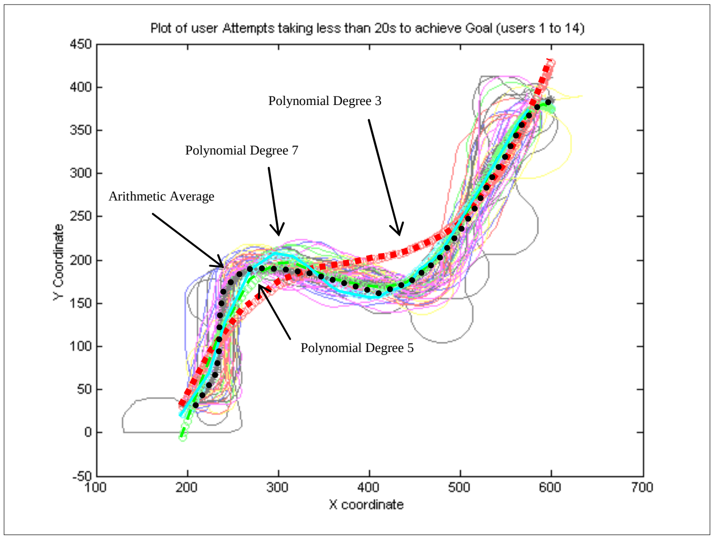


Figure 12: Plot of steady state user trajectories with the average trajectory and polynomial fits of degree 3, 5 and 7 superimposed.

7. Conclusions and Future Work

In this paper we have presented a unique historical perspective on the development of DIAs within the military and academic domains, noting also that research into networked games has made a significant contribution to the current state-of-the-art. We have defined latency and described a new approach to masking latency based on user strategy models. The details of a game-like application were presented. The tool offers a unique psychological window into strategies people adopt when confronted by an unknown maze environment with visibility restrictions.

Terminology associated with strategies, goals and trajectories was introduced. An illustrative example based on a single strategy and single goal was presented.

This paper is based on the PhD work of one of the authors. The next step in the research is to extend the data collection and analysis to goals with goal multiplicity index of 2 or more (two strategies that lead to the same goal). New methods of modeling the data will also be investigated. In addition research will focus on developing a protocol for efficient dissemination of user strategy information.

References

[i] Stephen R. Ellis, ‘What are Virtual Environments?’, *IEEE Computer Graphics and Applications*, 14, 1, (January 1994), 17-22.

[ii] James Calvin, Alan Dickens, Bob Gaines, Paul Metzger, Dale Miller and Dan Owen., ‘The SIMNET Virtual World Architecture’, *Proceedings of IEEE VRAIS*, 1998, 450-455.

iii Zied Choukair and Damien Retailleau, 'Integrating QoS to Collaborative Distributed Virtual Reality Applications', *Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-time Distributed Computing,* (2000), New Port Beach, CA, USA, 320-327.

iv Jauvane C. de Oliveira and Nicolas D. Georganas, 'VELVET: An Adaptive Hybrid Architecture for Very Large Virtual Environments', *IEEE International conference on Communications, ICC,* 4 (2002), 2491-2495.

v Richard C. Waters and John W. Barrus, 'The rise of shared virtual environments', *IEEE Spectrum*, March 1997.

vi Sandeep Singhal and Michael Zyda, *Networked Virtual Environments; Design and Implementation*, (New York: ACM Press, 1999), ISBN: 0201325578.

vii Aaron McCoy, Declan Delaney and Tomas Ward, 'Game-state fidelity across distributed interactive games', *ACM Crossroads*, Networking Issue 9.4 (Summer 2003), 4-9.

viii Jouni Smed, Timo Kaukoranta and Harri Hakonen, 'Aspects of Networking in Multiplayer Computer Games', *Proceedings of International conference on application and development of computer games in the 21st century,* (November 2001), China, 74-81.

ix Ernest H. Page & Roger Smith, 'Introduction to Military Training Simulations: A guide for discrete event simulationists', *Proceedings of the 1998 Winter Simulation Conference,* (1998), 53-60.

x Website on advanced distributed simulation: http://www.ait.nrl.navy.mil/5585/ADS/ADSMain.html

xi Bernard P. Zeigler, Doohwan Kim and Herbert Praehofer, *DEVS Formalism as a Framework for Advanced Distributed Simulation*, Proceedings of the 1st International Workshop on Distributed Interactive Simulation and Real-Time Applications (1997), 15-21.

xii See MITRE project website: http://www.mitre.org/

xiii IEEE Standard for Distributed Interactive Simulation - Application Protocols IEEE Std 1278.1-1995, (Revision of IEEE Std 1278-1993).

xiv Tzi-Cker Chiueh, 'Distributed Systems support for Networked Games', *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS VI),* (May 5-6, 1997), 99-104.

xv Judith S. Dahmann, Richard M. Fujimoto, and Richard M. Weatherly, 'The Department of Defense High Level Architecture', *Winter Simulation Conference* (December 1997).

xvi SISO website: http://www.sisostds.org/

xvii Jauvane C. de Oliveira, Shervin Shirmohammadi, Nicolas D. Georganas, *Collaborative Virtual Environment Standards: A performance Evaluation*, Proceedings of the 3rd IEEE International Workshop on Distributed Simulation and RealTime Applications (DS-RT '99) 23 - 24 October, 1999 College Park, Maryland.

xviii Chris Greenhalgh, *Large Scale Collaborative Virtual Environments*, (New York: Springer, 1999) ISBN: 1852331488.

xix Chris Greenhalgh, Purbick J. and Dave Snowdon, 'Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring', *Proceedings of the CVE 2000 Symposium,* (2000), 119-127.

xx Sandeep Kishan Sandeep, 'Effective Remote Modeling in Large-scale Distributed Simulation and Visualization environments', (unpublished PhD thesis, Stanford University, 1996).

xxi DIVE homepage: http://www.sics.se/dive/

xxii Jauvane C. de Oliveira et al (see reference iv)

xxiii Jauvane C. de Oliveira et al (see reference iv)

xxiv Andrzej Kapolka, Don McGregor and Michael Capps, 'A unified component framework for dynamically extensible virtual environments', *ACM CVE02,* (September 30th - October 2nd 2002), 64-71.

xxv Sun Microsystems. Java – JDK 1.4.0. 2002. http://java.sun.com/

xxvi Dongman Lee, Mingyu Lim and Seunghyun Han, 'ATLAS - A Scalable Network Framework for Distributed Virtual Environments', *ACM CVE02,* (September 30th - Oct 2nd 2002), 47-54.

xxvii Katherine L. Morse, Lubomir Bic and Michael Dillencourt, 'Interest Management in large-scale virtual environments', *Presence Teleoperators and virtual environments,* 9, (1), (February 2000), 52-68.

xxviii John W. Barrus, Richard C. Waters and David B. Anderson, 'Locales: Supporting Large Multi-user Virtual Environments', *IEEE Computer Graphics and Applications,* 16 (6), (1996), 50-57.

xxix Pedro Garcia, Oriol Montala, Carles Pairot, Robert Rallo and Antonio Gomez Skarmeta, 'MOVE: Component Groupware Foundations for Collaborative Virtual Environments', *ACM CVE02,* (September 30th - Oct 2nd 2002), 55-62.

xxx VRML - Virtual Reality Modeling Language web site: http://www.vrml.org/

xxxi H-Anim - specification for a standard VRML humanoid or avatar: http://www.h-anim.org/

xxxii Jauvane C. de Oliveira et al (see reference iv)

xxxiii Sandeep Singhal and Michael Zyda (see reference vi) xxxiv Juri Munkki, 'Design and Implementation of Networked Games', website: http://www.hut.fi/~jmunkki/netgames/, (December 15th 1997).

xxxv George J Valentino, Steve T. Thompson, Todd Kniola, 'An SMP-based, Low-latency, Network Interface Unit and Latency measurement System: the SNAPpy system', *Proceedings of the 2nd IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT '98)* (19 - 20 July, 1998) Montreal, Canada.

xxxvi Stuart Cheshire, 'Latency and the quest for interactivity', A White paper commissioned by Volpe Welty Asset Management, L.L.C., for the synchronous person-to-person interactive computing environments meeting. http://www.stuartcheshire.org/cheshire/papers/LatencyQuest.html, (November 1996).

xxxvii Paul M. Sharkey, Matthew D. Ryan and David J. Roberts, 'A local perception filter for Distributed Virtual Environments', *Proceedings of Virtual Reality Annual International Symposiun (VRAIS'98)* (March 1998), Atlanta, Georgia, USA, 242-249.

xxxviii Christopher Diot and Laurent Gautier, 'A Distributed Architecture for Multiplayer Interactive Applications on the internet', *IEEE Network* (July/August 1999), 6-15.

xxxix Kuo-Chi Lin and Daniel E. Schab, 'The performance assessment of the dead reckoning algorithms in DIS', *Simulation*, 63, (5) (November 1994) 318-325.

xl Mostafa A. Bassiouni, Ming-Hsing Chiu, Margaret Loper, Michael Garnsey and Jim Williams, 'Relevance filtering for distributed interactive simulation', *Computer Systems Science and Engineering*, 13, (1) (January 1998), 39-47.

xli Mostafa A. Bassiouni, Ming-Hsing Chiu, Margaret Loper, Michael Garnsey and Jim Williams, 'Performance and reliability analysis of relevance filtering for scalable distributed interactive simulation', *ACM Transactions on modeling and computer simulation*, 7, (3), (July 1997),293-331.

xlii Radharamanan Radhakrishnan and Philip A. Wilsey, 'Ruminations on the implications of Multi-Resolution modelling on DIS/HLA', *Proceedings of the 3rd IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT '99)*, (23 - 24 October, 1999) College Park, Maryland.

xliii Benjamin G. Worthington and David J. Roberts, 'Encapsulating Network Latency Compensators in VRML', *Proceedings of VWSIM (Virtual Worlds and Simulation Conference - part of SCS Western Multi-Conference 2000*, (2000) San Diego, USA.

xliv David J. Roberts and Paul M. Sharkey, 'Maximizing Concurrency and Scalability in a Consistent, Causal, Distributed Virtual Reality System, whilst minimizing the effect of Network Delays', *IEEE proceedings WETICE 1997: 6th Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, (June 1997), M.I.T. Boston, USA.

xlv J. Mark Pullen, 'Reliable Multicast network Transport for Distributed Virtual Simulation', *Proceedings of the 3rd IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT '99)*, (23 - 24 October 1999) College Park, Maryland.

xlvi Christopher Diot and Laurent Gautier, (see reference xxxiv)

xlvii Michael R. Macedonia and Michael Zyda, 'A Taxonomy for Networked Virtual Environments', *IEEE Multimedia*, 4 (1) (January-March 1997), 48-56.

xlviii Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker and Daniel Zappala, 'RSVP: A new Resource Reservation Protocol', *IEEE Communications Magazine*, 50th anniversary commemorative edition, (May 2002) 116-127. [Originally published in September 1993 in IEEE Network Magazine, Vol. 7, No. 5.]

xlix George V. Popescu and Christopher F. Codella, 'An architecture for QoS data replication in Network Virtual Environments', *Proceedings of the IEEE Virtual Reality*, (2002).

l Declan Delaney, Tomas Ward, Seamus McLoone, 'On Reducing Entity State Update Packets in Distributed Interactive Simulations using a Hybrid Model', *Proceeding of the 21st IASTED International Multi-conference on Applied Informatics* (10-13 February 2003), ISBN 0-88986-341-5, Innsbruck, Austria.

li The MathWorks, Inc. http://www.mathworks.com/

lii Corina Sas, Gregory M. P. O'Hare, Ronan Reilly, *On-line Trajectory Classification*, In Proceedings of International Conference on Computational Science 2003 (ICCS 03), Workshop on Scientific Visualisation and Human-Machine Interaction in a Problem-Solving Environment, (2nd-4th June 2003), Brisbane, Australia, LNCS, Springer Verlag Publishers.

APPENDIX A

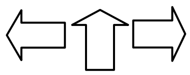
Universe Movement Mapper
Experimental Protocol for Data recording

Explanation of game given to each participant:

You are being asked to navigate a spacecraft through a maze from a given starting point. There is one target in the maze – (see icon below). Navigation is achieved by pressing the arrow keys. The arrow keys have the following functionality:

Move forward (keep pressed to accelerate to max velocity)

Rotate anti clockwise



Rotate clockwise

Target Icon

The maze is bounded by the blue rectangle delimiting the screen area. The maze layout, the starting position and the location of the target remain invariant. As you move about a short yellow trail illustrates the last few seconds of movement. The objective of the game is to reach the target in the shortest time possible and get your name on the Best Scores list. You may play the game as many times as you wish. Between each attempt the data will be saved to file. Between goes click the 'GO' button.

You will be given <u>ONE</u> trial run to test the dynamics of the spacecraft.

****

Name: Age:

Today's Date:

Male or Female Right handed ft handed

Do you play games? (Never) 0 1 2 3 4 5 6 7 8 9 10 (Often)

How many other times have you used this Universe Mapper program?

						Time Allowed (s)	Maze Name	Saved Files
Circle	10	50	100	150	200			
Square	10	50	100	150	200			
Cone	10	50	100	150	200			

<u>Comments:</u>