VISUAL SIMNET

PROJECT REPORT 2002-2003

SUBMITTED BY

C.KARTHIGA DEVI (9927K0127)
S.KAVITHA (9927K0132)
NISHI ANNA ABRAHAM (9927K0144)

Under the guidance of
Ms.S.Rajini
(Senior Lecturer, Department of Computer Science and Engg.)

in partial fulfillment of the requirement for the award of the degree of
Bachelor's Degree in Computer Science and Engineering
of Bharathiar University, Coimbatore - 641042

Estd. 1984

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore - 641006

1984

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE, TAMILNADU-641 006

ISO 9001:2000

Department Of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Report entitled

"VISUAL SIMNET"

is a bonafide record of work done by

C.Karthiga Devi	***(9927k0127)***
S.Kavitha	***(9927k0132)***
Nishi Anna Abraham	***(9927k0144)***

in partial fulfillment of the requirements for the award of the degree of
BACHELOR OF ENGINEERING - COMPUTER SCIENCE
of Bharathiar University, Coimbatore during the academic year 2002-2003.

....Rajini.....
(Head of the Department) ***(Project Guide)***

Submitted for the University examination held on : 17-03-2003

....
(Internal Examiner) **(External Examiner)**

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We would like to take this opportunity to express our sincere gratitude to all the people who have helped and guided us in the various stages of this project.

We would like to extend our deep gratitude to **Dr.K.K.Padmanabhan**, Principal, Kumaraguru College of Technology, Coimbatore for presenting us this opportunity to work on our project.and for extending constant support and valuable guidance throughout the project work.

We are extremely thankful to **Dr.S.Thangasamy**, Professor and Head of the Computer Science and Engineering Department for being a source of inspiration for our project.

Our heartiest thanks to **Ms.S.Rajini**, Senior Lecturer and internal guide for her valuable guidance and constant support during the course of the project.

We are extremely indebted to all our teachers, a special mention to our Class Advisor, **Mr.M.N.Guptha** and non-teaching staff of Kumaraguru College of Technology for the help and support they have extended to us.

Lastly, we would like to thank our family and friends for their love and moral support without which the succesful completion of the project would not have been possible.

SYNOPSIS

SYNOPSIS

Visual Simnet is a modeling, discrete event simulation and reachability graph analysis system. It aims to provide graphical and textual editor for animated simulation and helps us to find the mean and momentary values.

It is mainly used to produce an imitation of queuing network and to perform function, structure, capacity & performance examination and prediction. It reduces the calculation expenses which actually increase with model size. In addition to this, it provides an analytical model which can be tailored for specific problems and situations.

Visual SimNet supports modeling with a universal model description language and the graphical model representation, several features for debugging and validation of the model, it supports simulation with an efficient simulation engine and graphical result presentation as well as structural analysis with its reach ability graph generation.

Visual SimNet is intended for research, development and education within the fields of Petri Nets, modeling, simulation and structural analysis or as a tool supporting other science departments by means of these methods.

Modeling helps you to reduce the complexity; a computer model reduces costs in terms of time and money of a real world model. Sometime it prevents danger for persons, material or environment or overcomes regulatory restrictions. Simulation reduces the preparation time for different experiments, insures the replicatibility of experiments, and reduces the experiment execution time.

Visual SimNet uses normal ASCII files for its configuration information, which are placed within the SimNet directory. Therefore Simnet is very easy to install and uninstall. No special setup program is required for the installation process; the unzip procedured will suffice for this purpose. To uninstall, SimNet directory requires to be deleted.

CONTENTS

CONTENTS

TITLE	PAGE NO.
Acknowledgement	(i)
Synopsis	(ii)
Contents	(iii)
Chapter 1 Introduction	
1.1 Modeling and Simulation	1
1.2 Features of the Proposed System	3
1.3 Current Status	4
1.4 Relevance and Improtance	5
Chapter 2 Literature Survey	7
Chapter 3 Requirements	
3.1 Software Requirements	10
3.2 Functional Requirements	10
3.3 Interface Requirements	12
Chapter 4 Solution Strategy	13
Chapter 5 Design Details	
5.1 Phase-I	14
5.2 Phase-II	15
5.3 Phase-III	15
Chapter 6 Testing	
6.1 Objectives of Testing	16
6.2 Testing Methods	16
Conclusion	18
Future Outlook	20
Bibliography	21
Coding	22
Sample Screens	40

INTRODUCTION

MODELING AND SIMULATION

Modeling is a transformation from the original to essential characteristics, which are important to the aim of investigations. The resulting simplification allows investigations on the model, which with the original are uneconomical, time expensive or impossible due to complexity.

Simulation is used in the following to transform the essential characteristics of an artificial, dynamic system to a formal, mathematical model, where the original process is modeled by an algorithm of structure and time dependent value changes of the concerned state variables. Subject to the restrictions resulting from the abstraction, various degrees of analysis can be done independent from the real system.

Results from such a model can be achieved either by analysis or by simulation. There exist different modeling conceptions. They differ mainly by the chosen basic elements of which the model is composed. The definition of basic elements again is largely determined by the possible solution methods, because both things strongly depend on each other. Up to now mainly queuing systems and extended Petri Nets have been used.

For the modeling of discrete events there are many methods, at present with Petri Nets and Queuing Systems two basic techniques. Though the two methods have different roots, by numerous extensions to each in the past, they have become very similar to each other. Some research on integration of Queuing systems and Petri Nets has already been done. Also Visual SimNet is combining both methods.

Petri Nets are distinguishable from queuing systems by certain characteristics which are important for modeling of computer systems. They allow a better modeling of the mutual dependencies and synchronization of processes. A further advantage of Petri Nets is that they consist of a small number of basic elements.

Basic concepts of Petri–Nets:

A Petri-Net consists of **places** and **transitions**.

They are connected by directed **arcs**, which can transfer **tokens**.

The number of tokens, which is taken from or put to a place, is defined by the **weight** of an arc.

Token can stay only in places. They are used to represent states.

The number of tokens a place can keep, is called its **capacity**.

The number of tokens which are actually in a place is called **marking**.

The **pre-condition** means, that the number of tokens in the pre-located place must be >= the weight of the arc connecting this place and the transition.

The **post condition** means, that the number of tokens the post located place can take must be >= the weight of the arc connecting the transition and this place.

If and only if all pre- and post conditions of a transition are fulfilled, this transition gets concession and may **fire** (take tokens from pre-located places and put tokens to post located places).

In this way **events** (change of states) are represented by firing of a transition.

This allows one to get familiar with the modeling technique in a short time, as well as for simplicity of models and the simulation algorithm. For these reasons Petri Nets were used for Visual SimNet.

FEATURES OF THE PROPOSED SYSTEM:

Features which makes the proposed system different or sets it apart from the currently existing system are the following:

- ✓ **Editor with Multi document interface**
- ✓ **New syntax**
- ✓ **Hierarchical model**
- ✓ **Modeling elements definable and extensible**

CURRENT STATUS

Visual SimNet is a modeling, discrete event simulation and reach ability graph analysis system. It aims to provide graphical and textual editor for animated simulation and helps us to find the mean and momentary values. it is mainly used to produce an imitation of queuing network and to perform function, structure, capacity and performance examination and prediction. it reduces the calculation expenses which actually increases with model size and it also provides an analytical model which can be tailored for specific problems.

Features of the developed system:

- ✓ It has both graphical and textual model editor, automatic conversion in both directions.
- ✓ Animated simulation
- ✓ Measurement of mean and momentary values (waiting time, queue length, ...)
- ✓ Distribution analysis with recognition of distribution type and parameters.
- ✓ Variable parameter experiments.
- ✓ Selectable accuracy of results.
- ✓ Graphical and textual result presentation (curves, histograms, statistics).
- ✓ Reachability graph analysis.

RELEVANCE AND IMPORTANCE

All real world and theoretical problems require qualitative or quantitative statements. Modeling helps you to reduce the complexity, a computer model reduces costs in terms of time and money of a real world model. Sometime it prevents danger for persons, material or environment or overcomes regulatory restrictions. Simulation reduces the preparation time for different experiments, insures the replicatibility of experiments, and reduces the experiment execution time.

Visual SimNet supports modeling with an universal model description language and the graphical model representation, several features for debugging and validation of the model, it supports simulation with an efficient simulation engine and graphical result presentation as well as structural analysis with its reach ability graph generation. Visual SimNet is intended for research, development and education within the fields of Petri Nets, modeling, simulation and structural analysis or as a tool supporting other science departments by means of these methods.

Simulation has two essential advantages. First, calculation expense does not increase as fast with bigger models as with other methods. But this advantage can be used only with big models, because small models may be solved faster analytically if a solution method exists.

Second, analytical methods are usually tailored for specific problems, and they are valid only under certain conditions. But in practice different problems occur, usually combined systems. Then simulation is a universal and sometimes the only possible solution method to be followed.

It provides the imitation of the real time networks. All the nodes along with their transitions can be represented in the model. The model can be used for the future simulation. Thus, this system helps us both in modeling and simulating a real time network which is actually available only
in different systems till now. This actually tells us about the problems that will occur while implementing a proposed network without direct implementation in the real world.

It is mainly developed with an aim of simulating the real time networks. The simulations are actually implementing using the algorithms. The simulation results are actually given both in textual mode and in graphical mode. The simulation process are mainly done on the modeled network

Relations between state variables, parameters, start conditions and input

information is described in the form of analytical (functional) terms (algebraic, integral, differential and difference equations). The required results can be achieved by analytical transformation or by analytical methods.

Analytical solutions exist for queuing systems as well as for Petri Nets. But complex systems often can not be expressed as an analytical model or if it can there aren't analytical or numerical methods for its solution. Because the calculation cost for analytical models increases exponential with the model size, the calculation of complex models also meets with technical limits.

The structural analysis is based on the consideration of all possible states of a system, as well as of the transitions between them. Thus e.g. locks, forbidden states, dead substructures and the maximum utilization of queues may be discovered. In the following mainly the characteristics are described, which may be examined by the reach ability graph analysis based on Petri Net models. Reach ability graph analysis is supported by Visual SimNet.

Visual SimNet uses normal ASCII files for its configuration information, which are placed within the SimNet directory. Therefore Simnet is very easy to install and uninstall. No special setup program is required; unzip procedures will suffice for the installation process. To uninstall, the SimNet directory requires to be deleted.

LITERATURE SURVEY

LITERATURE SURVEY

The Fluid Stochastic Petri Net simulator

Authors: ***D.M. Nicol, A.S. Miner,*** Dept. of Computer Science, College of William & Mary, Williamsburg, VA, USA

Abstract:

A Fluid Stochastic Petri Net (FSPN) extends a normal Petri net with the notion of fluid places or nodes and fluid arcs. Fluid levels are continuous, and may be used to approximate the presence of many discrete tokens. We describe an FSPN simulation tool whose input description extends the syntax of the widely used tool spnp.

Petri-Net-Based Modeling and Evaluation of Pipelined Processing of Concurrent Database Queries

Authors: ***K.P.Mikkilineni, Y.C.Chow, S.Y.W. Su***

Abstract:

A description is given of a Petri-net-based methodology for modeling and evaluation of pipelined processing of concurrent database queries in an integrated data network (IDN). An extended Petri-net model is presented and used to model two key approaches to concurrent database query processing in the IDN, namely, pipelined and data-flow-based execution of queries and intermediate data sharing among concurrent queries. Database operations are categorized, and the models for the data flow and control flow in them are presented. A general-purpose Petri-net simulator has been developed using event-driven programming techniques and used to simulate the execution of the Petri-net models of some test queries. The results validate the results of a previous analytical evaluation in which the advantages of pipeline and intermediate data sharing were established. Since all the essential details of query processing in the IDN have been simulated, the results of this simulation study are believed to present closely the workings of the actual system.

A Real Delay Switching Activity Simulator based on Petri Net Modeling

Authors: ***Ashok K. Murugavel, N.Ranganathan,*** University of South Florida

Abstract:

Switching activity estimation is an important step in power estimation of digital VLSI circuits. While simulation yields accurate results, it is time consuming. In this paper, we propose a new technique based on Petri nets for real-delay switching activity estimation that yields the same accuracy as simulation, but is significantly faster in computation. We introduce a new type of Petri net called Hierarchical Colored Hardware Petri Net (HCHP-Net). The gate-level circuit is first transformed into a directed acyclic graph called, GSDAG, in which both the gates as well as the signals correspond to the nodes in the graph. The GSDAG is then mapped onto a corresponding HCHP-Net which is then simulated using a Petri net simulator. Experimental results for ISCAS '85 circuits are presented. The method replicates exactly the switching activity results for real-delay models produced by HSPICE and PowerMill. However, the per-pattern simulation time is about 51 times faster than the Synopsys PowerMill and 8900 times faster than the Avanti HSPICE

A Comparison on Neural Net Simulators

Authors: ***Ottmar Lutzy, Andreas Dengel***

Abstract:

Five artificial neural net simulators that are free of charge, run on Sun Sparcstations under X windows, and are available by anonymous FTP over the Internet, have been tested. The simulators are PlaNet version 5.6 from the University of Colorado at Boulder, Pygmalion version 2.0 from University College in London, Rochester Connectionist Simulator version 4.2 from the University of Rochester, and Stuttgart Neural Net Simulator versions 1.3 and 2.0 from the Institute for Parallel and Distributed High-Performance Systems at the University of Stuttgart in Germany. To compare the simulator's functionality, the number as well as the range of facilities they offer to generating and working with connectionist networks are considered. One artificial neural net was generated with the same topology and functions for all the simulators. A three-layered feedforward net with a backpropagation learning function was used for recognizing printed characters. The test results are discussed.

Simulation Study of Interoperative Methods for Congestion Control over TCP/AT

Authors: ***Pil-Joong Kim, Yeon Lee, Woon-Sik Kim, Kwang-Il Lee, Sang-Ha Kim*** , Dept. of Computer Science, Konyang University, Nonsan, South Korea

Abstract:

ATM rate-based flow controls can effectively reduce cell losses and buffer requirement

in ATM switches, and thus enhance overall performance in ATM networks. However, many simulation studies show that the performance of TCP can be reduced over ATM networks in some situation like network congestion. This comes from the absence of direct channel between TCP and ATM protocol stack. That is, the ATM layer in a host does not notify TCP of the congestion information from ATM networks through resource management (RM) cells. So, TCP may generate vast packets regardless of present network status. In this paper, we introduce the direct mechanism, interoperative method between TCP and ATM protocol stack and perform a simulation in order to justify our mechanism. The simulation results, by SIMNET from NIST, show that the TCP adopting mechanism has better performance over that only with ATM congestion control mechanism.

TimeNET-Sim-a parallel simulator for stochastic Petri Nets

Authors: ***C. Kelling,*** Institute for Tech. Information, Tech. University. Berlin, Germany

Abstract:

TimeNET is a software package for modeling and performance evaluation with non-Markovian Petri nets. Concepts and implementation of the simulation component of this tool are introduced. The paper focuses on a reliable statistical analysis and the application of variance reduction techniques in a parallel, distributed simulation framework. It examines the application of variance reduction with control variates and shows an approach for their automatic selection in Petri net models. An example demonstrates the gain of variance reduction and parallelization.

Estimating the Number of Faults using Simulator based on Generalized Stochastic Petri-Net Model

Authors: ***Osamu Mizuno, Shinji Kusumoto, Tohru Kikuno,*** Osaka University
Yasunari Takagi, Keishi Sakamoto, OMRON Corporation, Japan

Abstract:

In order to manage software projects quantitatively, we have presented a new model for software project based on Generalized Stochastic Petri-net model which can take influence of human factors into account, and we have already developed software project simulator based on GSPN model. This paper proposes methods for calculating model parameters in the new model and estimating the number of faults in the design and debug phases of software process. Then we present experimental evaluation of proposed method using a data of actual software development project on a certain company. As the result of case study, we confirmed its effectiveness with respect to estimating the number of faults in the software process.

REQUIREMENTS

SOFTWARE REQUIREMENTS:

Definition:

Visual SimNet is a modeling, discrete event simulation and reachability graph analysis system, based on Petri Nets.

User characteristics:

The end user need not specifically be an expert in any particular software. The system is very user-friendly since all the operations included in the system is menu-driven.

User problem statement:

The current system has the problem of using separate editor for modeling, evaluation and research. But here we intent to provide all the features in a single editor document.

FUNCTIONAL REQUIREMENTS:

Graphical and Textual Model Editor

- The system has an editor which accepts both graphical and textual information
- This feature is essential to make the system more user interface
- Editor will support multi document interface

Animated Simulation

- The simulation information that can be interpreted from the system can be represented in form flowcharts, graphs, etc according to the user requirements.
- This feature enables the user to get the result in the format he so desires.

Measurement of mean and momentary values

- The momentary values are also calculated since this system supports Petri nets with time
- This feature is essential to help in producing the performance analysis and structural analysis.

Distribution analysis with Recognition of Distribution Type and Parameters

- The distribution analysis helps as to produce distribution of the various parameters with respect to time
- This feature helps the user to get the desired level of accuracy.

Variable parameter experiments

- Variable parameter experiments helps us to produce the structural and performance analysis.

Selectable accuracy of results

- The results can be calculated based on the level of accuracy and conciseness required by the user. The requirement of the correctness or preciseness of the solution varies with the whim and fancy of the user, the results are chosen acccordingly.

Reachability graph analysis

- A study on reachability graphs can be made accordingly and the various cases and situations can be analysed systematically.

INTERFACE REQUREMENTS:

User interface:

- The system is menu-driven. All modeling techniques are given as menu choices
- The system utilizes application programming interface function to provide public interface
- The system has a set of diagnostic tools and provides help facilities

Software interface:

Requires windows 9x/NT
Source Code:Visual C++

Performance speed:

Simulation speed : 6000 events/sec with Pentium 90 Mhz
Maximum model size : 128 000 elements

Non functional attributes:

- Security
- Reliability
- Maintainability
- Portability
- Extensibility
- Reusability

SOLUTION STRATEGY

SOLUTION STRATEGY

We attempt to develop an analysis system which will support both modeling and simulation of a real time network. This has to be implemented in GUI (Graphical user interface). This project is implemented using Microsoft Visual C++ 6.0.

The main reasons that we have considered for implementing our project in VC++ are as follows:

- ✓ Microscoft Visual C++ is a complete Windows application systems in one product.
- ✓ Visual C++ also includes ActiveX Template Library,which can be used to develop ActiveX controls for the internet.
- ✓ Docking window, configurable toolbars, plus customizable editors can easily be developed using Visual C++
- ✓ Visual C++ components and gallery helps in sharing software components among different projects
- ✓ Visual C++ contains a number of useful diagnostic tools.

DESIGN DETAILS

DESIGN DETAILS:

Phases of the Project:

- ✓ Editor Design along with tools
- ✓ Modeling Techniques Implementation
- ✓ Reports Generation.

Editor design with tools:

This phase mainly deals with the design of the graphical editor on which we can represent the network model. The editor is hereby provided with tools which help in representing the networks .The editor is menu driven. All modeling techniques and modeling structures are provided as menu choices and labeled buttons in the editor. The editor also provides a set of diagnostic tools and provides help facilities.

Editor Creation: The editor has the following options which are given as the top level menus.

File→ New
Open
Save
Save As
Print Preview
Print
Exit

Edit→ Undo
Cut
Copy
Paste
Clear

Reports→ Text
Graph

Test→ Queue length cal
Shortest path cal
Capacity determination
Evaluation of weight &
Marking.

Help→ Help f1
About Visual Simnet

A tool bar which represents the modeling structures are also given in the editor. Modeling structures are used to represent the network components like nodes, cables, routers, T-connectors, Hub, etc. We also provide a text box option using which we can also include some text info in the graphical representation.

Modeling Techniques Implementation:

- ✓ **Capacity Calculation:** It determines the number of tokens that the node can hold.
- ✓ **Shortest path calculation**: Given the source and destination, it determines the shortest path for the token to travel to reach the destination.
- ✓ **Queue length calculation:** It determines the number of tokens in the arc at the given time.
- ✓ **Evaluation of weight and marking:** It determines the number of tokens that are taken from or put to node at a given time (weight) & the number of tokens actually present at the given time in the node (marking).

Report generation:

Report generation is done both in the textual mode and graphical mode, depending on the user's choice.

In the graphical mode, the results can be represented in the form of cumulative frequency curves, histograms, etc. In cases where the output cannot be represented as graphs, the resultant paths are highlighted in the model.


TESTING

TESTING AND IMPLEMENTATION

The implementation phase is less creative then system design. No software is assumed completed until it is successfully tested and implemented.

An elaborate test data is prepared and the system is tested using the test data. While testing, errors are noted and corrections are made. The corrections are note for future use. Both the hardware and software securities are made to run the developed system successfully in future.

Objectives of testing:

- ✓ Testing is the process of executing a program with the intent of finding an error
- ✓ A good test case is one that has the high probability of finding yet undiscovered error
- ✓ A successful test is one that uncovers and yet undiscovered error

System testing is the stage of implementation which is aimed at ensuring that the system works accurately and effectively before the live operation commences. Testing is vital to success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

The candidate system is subjected to variety of tests online response volume, stress, recovery, and security and usability tests. A series of testing are preformed before the system is read for the user acceptance testing.

Types of testing:

The different ypes of testing are as follows:

- ✓ Unit testing
- ✓ Integrated testing
- ✓ Black box testing
- ✓ User acceptance testing

Unit testing:

In this testing each sub module is tested individually and integrated with the overall system. Unit testing focuses verification effort on the smallest unit software design of the module .This is also known as module testing. The modules of the system are tested separately. This testing was carried out during programming stage itself. In this testing step each module is found to be working satisfactorily as regard to the expected output from the module.

Integrated testing:

Data can be lost across an interface; one module can have an effect on another module, sub functions when combined may not produce the desired major function. Integrated testing is a systematic testing for constructing the program structure. This testing was done with the sample data. The developed system was done successfully with the test data. The need for the integrated test its find the overall system performance .The objective is to take unit tested module and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of course. Thus in the integrated step all errors are uncovered, corrected for the next testing steps.

Black box testing:

Black box testing is done in order to find

- ✓ Incorrect or missing functions
- ✓ Interface error
- ✓ Errors in external device access
- ✓ Performance error
- ✓ Initialization and termination errors

The above testing was successfully carried out for this system

Validation testing:

As the culmination of the black box testing proceeds, the software is completely assembled as a package, the various interfacing errors have been uncovered and corrected and the final series of software tests begins. Validation succeeds when the software functions in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of the two possible conditions exists.

- ✓ The function or performance characteristics confirmed to specification and are accepted
- ✓ The deviation from the specification is uncovered and deficiency list is created

Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

CONCLUSION

CONCLUSION

The framework for modeling and simulating a network is a simple and secure process which ensures full performance. The implementation of the project was simple, making use of the user friendly and GUI language, Visual C++ 6.0. The usage has been proved to be simple even allowing a person who is not familiar with graphical interface to work with this system.

FUTURE OUTLOOK

FUTURE ENHANCEMENTS

The project is limited in terms of reduction of costs and execution time. The future enhancements that can be done to enhance the efficiency and overall working of this project are listed down below:

- ✓ Additional extension of standard Petri nets are proposed to be added. This will reduce the expense and calculation time to a great extent.

- ✓ This can be extended to other fields like testing of an algorithm, modeling a business application and a number of similar or related applications.

BIBLIOGRAPHY

REFERENCES

- "Microsoft Windows Programmer's Reference", published by Microsoft Press

- "Database Programming with Visual C++", authored by Lyn Robison, published by Sams Publishing by arrangement with Macmillan Computer Publishing, U.S.A.

- "Programming with Visual C++", authored by David.J.Kruglinski, George Shepherd, Scott Wingo

- "Teach Yourself Visual C++", authored by David.A.Holzgang

CODING

// ChildView.cpp : implementation of the CChildView class

```
#include "stdafx.h"
#include "Nsimmodal.h"
#include "ChildView.h"

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

// CChildView

CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView,CWnd )
        //{{AFX_MSG_MAP(CChildView)
        ON_WM_PAINT()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


// CChildView message handlers

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
        if (!CWnd::PreCreateWindow(cs))
                return FALSE;

        cs.dwExStyle |= WS_EX_CLIENTEDGE;
        cs.style &= ~WS_BORDER;
        cs.lpszClass =
AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
                ::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1),
NULL);

        return TRUE;
}
```

```
void CChildView::OnPaint()
{
        CPaintDC dc(this); // device context for painting


        // Do not call CWnd::OnPaint() for painting messages
}
```

// MainFrm.cpp : implementation of the CMainFrame class

```
#include "stdafx.h"
#include "Nsimmodal.h"

#include "MainFrm.h"

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

// CMainFrame


BEGIN_MESSAGE_MAP(Mwind,CFrameWnd)
ON_COMMAND(ID_STARTEDIT,OnConnect)
ON_COMMAND(ID_NETWORKOPERATIONS_EXIT,Onstop)
ON_COMMAND(ID_HELP,Onhelp)
END_MESSAGE_MAP()

void Mwind::OnConnect()
        {

                Ndialog md(IDD_DIALOG3);
                md.DoModal();


        }
```

```
Mwind::Mwind()
        {
                CMenu *m;
                m=new CMenu();
                m->LoadMenu(IDR_MENU1);
                Create(0,"Network Simulator System");
                SetMenu(m);

        }

void Mwind::Onhelp()
        {

                Ndialog md(IDD_DIALOG1);
                md.DoModal();


        }


 void Mwind::Onstop()
        {
                AfxMessageBox("Thanks for using System");
        SendMessage(WM_CLOSE);
        }
 // CMainFrame diagnostics

 // CMainFrame message handlers
```

// Nsimmodal.cpp : Defines the class behaviors for the application.

```
//

#include "stdafx.h"
#include "Nsimmodal.h"
#include "MainFrm.h"

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

// CNsimmodalApp

class Mdialog:public Cdialog
{
```

```
public:
        Mdialog(int n):CDialog(n)
        {

        }

};

BEGIN_MESSAGE_MAP(CNsimmodalApp, CWinApp)
        //{{AFX_MSG_MAP(CNsimmodalApp)

        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


// CNsimmodalApp construction

CNsimmodalApp::CNsimmodalApp()
{
}

// The one and only CNsimmodalApp object

CNsimmodalApp theApp;

// CNsimmodalApp initialization

BOOL CNsimmodalApp::InitInstance()
{
        AfxEnableControlContainer();

        // Standard initialization

#ifdef _AFXDLL
        Enable3dControls();                     // Call this when using MFC in a shared
DLL
#else
        Enable3dControlsStatic();       // Call this when linking to MFC statically
#endif

                Mwind *mw;
                mw=new Mwind;
                m_pMainWnd=mw;
                Mdialog md(IDD_DIALOG2);
                md.DoModal();
```

```
            m_pMainWnd->ShowWindow(SW_MAXIMIZE);
            m_pMainWnd->UpdateWindow();
            return TRUE;


        return TRUE;
}

// CNsimmodalApp message handlers

// CAboutDlg dialog used for App About

// CNsimmodalApp message handlers

// ChildView.h : Interface of the CChildView class
//

#if
!defined(AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__IN
CLUDED_)
#define
AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__INCLUDED
_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// CChildView window

class CChildView : public CWnd
{
// Construction
public:
        CChildView();

// Attributes
public:

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CChildView)
```

```
protected:
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CChildView();

        // Generated message map functions
protected:
        //{{AFX_MSG(CChildView)
        afx_msg void OnPaint();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__IN
CLUDED_)
```

// MainFrm.h : interface of the CMainFrame class

```
//

#if
!defined(AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INC
LUDED_)
#define
AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ChildView.h"

#include<afxwin.h>
#include"resource.h"
```

```
class Ndialog:public CDialog
{


public:
        Ndialog(int n):CDialog(n)
        {

        }

};

class Mwind:public CFrameWnd
{

public:

        Mwind();
        afx_msg void OnConnect();
        afx_msg void Onhelp();
        afx_msg void Onstop();

                DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INC
LUDED_)
```

```
// Nsimmodal.h : main header file for the NSIMMODAL application
//

#if
!defined(AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__I
NCLUDED_)
#define
AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__INCLUDE
D_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
#include "Mainfrm.h"

// CNsimmodalApp:
// See Nsimmodal.cpp for the implementation of this class
//

class CNsimmodalApp:public CWinApp
{

public:

        CNsimmodalApp();
        BOOL InitInstance();
DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__I
NCLUDED_)
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Nsimmodal.rc
//
#define IDD_DIALOG1                 101
#define IDR_MENU1                   102
#define IDD_DIALOG2                 103
#define IDB_BITMAP1                 104
#define IDD_DIALOG3                 105
#define IDC_NSIMULATOR1             1000
#define ID_NETWORKOPERATIONS_CONNECTSYSTEM 40001
#define ID_STARTEDIT                40001
#define ID_NETWORKOPERATIONS_EXIT   40002
#define ID_HELP_ABOUTSYSTEM         40003
#define ID_HELP_HELPTOPICS          40004
#define ID_REQ                      40005

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE    107
#define _APS_NEXT_COMMAND_VALUE     40006
#define _APS_NEXT_CONTROL_VALUE     1001
#define _APS_NEXT_SYMED_VALUE       101
#endif
#endif
```

// stdafx.h : include file for standard system include files, or project specific include
//files that are used frequently, but are changed infrequently
//

```
#if
!defined(AFX_STDAFX_H__A8AF868B_07AB_46BD_A3D6_344D23FDFCC2__INC
LUDED_)
#define
AFX_STDAFX_H__A8AF868B_07AB_46BD_A3D6_344D23FDFCC2__INCLUDED
_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
```

```
#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>           // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_STDAFX_H__A8AF868B_07AB_46BD_A3D6_344D23FDFCC2__INC
LUDED_)
```

// ChildView.h : interface of the CChildView class

```
//

#if
!defined(AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__IN
CLUDED_)
#define
AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__INCLUDED
_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// CChildView window

class CChildView : public CWnd
{
// Construction
public:
        CChildView();

// Attributes
```

```
public:

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CChildView)
        protected:
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CChildView();

        // Generated message map functions
protected:
        //{{AFX_MSG(CChildView)
        afx_msg void OnPaint();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_CHILDVIEW_H__74D16E95_D3FE_43F5_80EB_761E99FF9EE4__IN
CLUDED_)
```

// ChildView.cpp : implementation of the CChildView class

```
//

#include "stdafx.h"
#include "Nsimmodal.h"
#include "ChildView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// CChildView

CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}


BEGIN_MESSAGE_MAP(CChildView,CWnd )
	//{{AFX_MSG_MAP(CChildView)
	ON_WM_PAINT()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


// CChildView message handlers

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
	if (!CWnd::PreCreateWindow(cs))
		return FALSE;

	cs.dwExStyle |= WS_EX_CLIENTEDGE;
	cs.style &= ~WS_BORDER;
	cs.lpszClass =
AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
		::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1),
NULL);

	return TRUE;
}

void CChildView::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	// Do not call CWnd::OnPaint() for painting messages
}
```

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "Nsimmodal.h"

#include "MainFrm.h"

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

// CMainFrame


BEGIN_MESSAGE_MAP(Mwind,CFrameWnd)
ON_COMMAND(ID_STARTEDIT,OnConnect)
ON_COMMAND(ID_NETWORKOPERATIONS_EXIT,Onstop)
ON_COMMAND(ID_HELP,Onhelp)
END_MESSAGE_MAP()

void Mwind::OnConnect()
        {

                Ndialog md(IDD_DIALOG3);
                md.DoModal();

        }

Mwind::Mwind()
        {
                CMenu *m;
                m=new CMenu();
                m->LoadMenu(IDR_MENU1);
                Create(0,"Network Simulator System");
                SetMenu(m);
        }

void Mwind::Onhelp()
        {
                Ndialog md(IDD_DIALOG1);
                md.DoModal();
        }
```

```
void Mwind::Onstop()
        {
                AfxMessageBox("Thanks for using System");
        SendMessage(WM_CLOSE);
        }

// CMainFrame diagnostics

// CMainFrame message handlers

// Nsimmodal.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Nsimmodal.h"

#include "MainFrm.h"

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

// CNsimmodalApp

class Mdialog:public CDialog
{


public:
        Mdialog(int n):CDialog(n)
        {

        }

};

BEGIN_MESSAGE_MAP(CNsimmodalApp, CWinApp)
        //{{AFX_MSG_MAP(CNsimmodalApp)

        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// CNsimmodalApp construction
```

```
CNsimmodalApp::CNsimmodalApp()
{
}

// The one and only CNsimmodalApp object

CNsimmodalApp theApp;

// CNsimmodalApp initialization

BOOL CNsimmodalApp::InitInstance()
{
        AfxEnableControlContainer();

        // Standard initialization

#ifdef _AFXDLL
        Enable3dControls();                     // Call this when using MFC in a shared
DLL
#else
        Enable3dControlsStatic();    // Call this when linking to MFC statically
#endif

                Mwind *mw;
                mw=new Mwind;
                m_pMainWnd=mw;
                Mdialog md(IDD_DIALOG2);
                md.DoModal();
                m_pMainWnd->ShowWindow(SW_MAXIMIZE);
                m_pMainWnd->UpdateWindow();
                return TRUE;




        return TRUE;
}

// CNsimmodalApp message handlers

// CAboutDlg dialog used for App About

// CNsimmodalApp message handlers
```

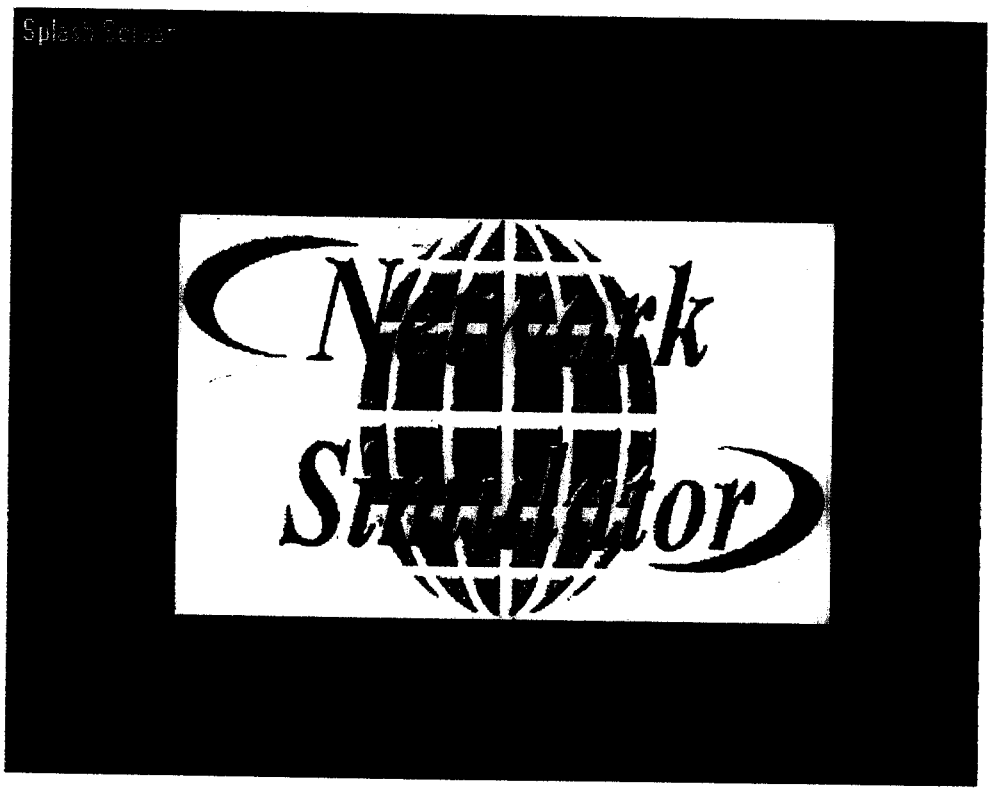
```
// MainFrm.h : interface of the CMainFrame class
//

#if
!defined(AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INC
LUDED_)
#define
AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ChildView.h"

#include<afxwin.h>
#include"resource.h"

class Ndialog:public CDialog
{

public:
        Ndialog(int n):CDialog(n)
        {

        }

};

class Mwind:public CFrameWnd
{

public:

        Mwind();
        afx_msg void OnConnect();
        afx_msg void Onhelp();
        afx_msg void Onstop();

                DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.
```

```
#endif //
!defined(AFX_MAINFRM_H__1E4B9F3E_8B49_47DD_A94E_953C21B88E29__INC
LUDED_)


// Nsimmodal.h : main header file for the NSIMMODAL application
//

#if
!defined(AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__I
NCLUDED_)
#define
AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__INCLUDE
D_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols
#include "Mainfrm.h"

// CNsimmodalApp:
// See Nsimmodal.cpp for the implementation of this class
//

class CNsimmodalApp:public CWinApp
{
public:

        CNsimmodalApp();
        BOOL InitInstance();
DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.
#endif //
!defined(AFX_NSIMMODAL_H__61BB2B71_46C9_433D_81FD_6DF12697E6F7__I
NCLUDED_)
```

VB ACTIVEX CONTROL

```
Private Sub Command1_Click()
DoEvents
Shell App.Path + "\nsimulator.exe", vbNormalFocus
DoEvents
SendKeys "{tab}", True
SendKeys "^{home}", True
SendKeys "^+{end}", True
SendKeys "{del}", True
SendKeys "^N", True
End Sub
```

SAMPLE SCREENS


Splash Screen