Grapher

Volume II

Submitted in partial fulfillment of the requirement
for the degree
of

MASTER OF COMPUTER APPLICATIONS

by

Sanjeev Kumar Guleria

MCA-25/91

Department of Computer Science and Engineering
Thapar Institute of Engineering and Technology, Patiala.
(Deemed to be a University)

1994

TABLE OF CONTENTS

```
/*****************************************************************
**********
 *
```

Grapher.h

```
   File Name:    Grapher.h
   This file contains global definitions for the Grapher
application.

                          *

*****************************************************************
*********/
#define LIST_SEPARATOR          ","
#define ITEM_SEPARATOR                " "
#define NODE_ID              1
#define PARENT_RELATION   2
#define CHILD_RELATION       3
#define PROCESS              4
#define VIEW_INFO            5
#define LAST_INFO            VIEW_INFO

#define RELATION_ID            1
#define RELATION_POS          2
#define OTHER_RELATION_ID     3
#define RELATION_NAME         4

#define NODE_VIEW_TYPE         1
#define NODE_VIEW_RECT         2
#define NODE_VIEW_NAME         3

#define CURSOR_PARENTCHILD   12
#define CURSOR_GRAPHER    13

#define GRAPHER_BASE 18000
#define ABOUTBOXRid     GRAPHER_BASE    /* Resource id for
the about box */
#define GETTEXTBOXRid   GRAPHER_BASE+1    /* Resource id for
the GetText box */
#define GETRELATIONBOXRid GRAPHER_BASE+2    /* Resource id
for the GetRelation box */

#define ABOUTText1      GRAPHER_BASE+51    /* Resource ids
for the about box's text ... */
#define ABOUTText2      GRAPHER_BASE+52
```

```
#define ABOUTText3      GRAPHER_BASE+53
#define GETText1        GRAPHER_BASE+54
#define GETRelationDlgText1   GRAPHER_BASE+55
#define GETRelationDlgText2   GRAPHER_BASE+56

#define MENUBarRid   GRAPHER_BASE+101      /* Resource id
for menu bar*/

#define OBJECTMenuRidGRAPHER_BASE+200      /* Resource id
for "Object" Menu*/
#define OBJECT_MODIFY   GRAPHER_BASE+201      /* Resource id
for "Modify" Menu item*/
#define OBJECT_DESTROY  GRAPHER_BASE+202      /* Resource id
for "Destroy" Menu item*/
#define OBJECT_ASSOCIATE   GRAPHER_BASE+203       /*
Resource id for "Associate" Menu item*/
#define OBJECT_EXECUTE  GRAPHER_BASE+204        /* Resource
id for "Execute" Menu item*/
#define OBJECT_SOCIALIZE   GRAPHER_BASE+205       /*
Resource id for "Socialize" Menu item*/
#define OBJECT_PARENT   GRAPHER_BASE+206      /* Resource id
for "Parent" Menu item*/
#define OBJECT_CHILD    GRAPHER_BASE+207      /* Resource id
for "Child" Menu item*/

#define OPTIONMenuRidGRAPHER_BASE+210      /* Resource id
for "Option" Menu*/
#define OPTION_MOVE     GRAPHER_BASE+211      /* Resource id
for "Move" Menu item*/
#define OPTION_AUTO     GRAPHER_BASE+212      /* Resource id
for "Auto" Menu item*/
#define OPTION_RELATION GRAPHER_BASE+213      /* Resource id
for "Relation" Menu item*/
#define OPTION_NODE_INFO GRAPHER_BASE+214       /* Resource
id for "Node Information" Menu item*/

#define VIEWMenuRid     GRAPHER_BASE+220      /* Resource id
for "View" Menu*/
#define VIEW_NONE       GRAPHER_BASE+221      /* Resource id
for "None" Menu item*/
#define VIEW_BUTTON     GRAPHER_BASE+222      /* Resource id
for "Button" Menu item*/
#define VIEW_GRAPH_RECTANGLE GRAPHER_BASE+223
#define GETTextEdit     GRAPHER_BASE+301
#define GETRelationDlgEdit1  GRAPHER_BASE+302
```

```
#define GETRelationDlgEdit2  GRAPHER_BASE+303

#define LISTBOXId  GRAPHER_BASE+401

#define SKETCHcmd    GRAPHER_BASE+501
#define NEWObjectCmd GRAPHER_BASE+502
#define DELETEObjectCmd GRAPHER_BASE+503
#define NEWNodeRelationCmd  GRAPHER_BASE+504
#define UPDATERelationInfoCmd  GRAPHER_BASE+505
#define GETOtherRelationIdCmd  GRAPHER_BASE+506
#define GETRelationNameCmd     GRAPHER_BASE+507
#define GETNodeTitleCmd     GRAPHER_BASE+508
#define FLIPNodeRelationCmd GRAPHER_BASE+509
#define UPDATEViewCmd GRAPHER_BASE+510
#define CLEARAllInfoCmd GRAPHER_BASE+511
#define SENDRelationCmd GRAPHER_BASE+512
#define KILLRelationCmd GRAPHER_BASE+513
#define VIEWRelationCmd  GRAPHER_BASE+514
#define LISTBOXSingleClickCmd GRAPHER_BASE+515
#define UPDATEProcessCmd GRAPHER_BASE+516
#define SETNodeInfoModeCmd GRAPHER_BASE+517

#define RELATION_VIEW_ID GRAPHER_BASE+900
#define RELATION_SHOW GRAPHER_BASE+901

#define PROCESS_FUNCTION GRAPHER_BASE+1000
#define PROCESS_EXEC GRAPHER_BASE+1001

#define GRAPHER_FILE "GRAPHER"
#define GRAPHER_TYPE    "GRF"
```

Grapher.url

```
/*
    Grapher.url: Example Universal Resource Language File
for XVT-Power++ applications.  You may add any application
specific resources.
*/
/*UNITS CHARS*/
#define APPNAME Grapher
#define QAPPNAME "Grapher"

#include "url.h"
#include "PwrURL.h"

#include "Grapher.h"

/* Definition of the about box dialog box: */

DIALOG ABOUTBOXRid , 100,100,390,120 /*,16, 7, 65, 9*/
   BUTTON DLG_CANCEL,   330, 80, 35, 30/*55, 6, 6, 2 */ "OK"
DEFAULT
   TEXT ABOUTText1      15, 15, 360, 15/*3, 1, 60, 1*/ "**
Grapher: An XVT-Power++ Application **"
   TEXT ABOUTText2      15, 40, 360, 15/*3, 3, 60, 1*/ "This
program illustrates Power programming"
   TEXT ABOUTText3      15, 60, 360, 15/*3, 5, 60, 1*/ "and
is part of Guleria's invasion into XVT-Power++."

/* Definition of GetText dialog box: */

DIALOG GETTEXTBOXRid,   130, 60, 205, 80/*11, 5, 31, 6*/ ""
MODAL
  TEXT GETText1       12,   15,  70, 15/*  ,1, 1, 9, 1*/
"Enter Name:"
  EDIT GETTextEdit,  82,   8, 105, 15/*10, 0, 19, 1*/
  BUTTON DLG_OK,  34,  40,  60, 30/*6, 3, 7, 3*/ "OK"
DEFAULT
  BUTTON DLG_CANCEL,  110, 40, 60, 30/*18, 3, 7, 3 */
"CANCEL"

/* Definition of dialog box: */

DIALOG GETRELATIONBOXRid,       130, 60, 260, 100."" MODAL
  TEXT GETRelationDlgText1       10,   12,  115, 15 "First
Relation Name:"
```

```
  EDIT GETRelationDlgEdit1,      135,   12, 100, 15
  TEXT GETRelationDlgText2       10,   36,  115, 15 "Second
Relation Name:"
  EDIT GETRelationDlgEdit2,      135,   36, 100, 15
  BUTTON DLG_OK,  110,  65,  40, 30 "OK"
/* Define the application menu bar: */

MENUBAR MENUBarRid

MENU MENUBarRid
   DEFAULT_FILE_MENU
   SUBMENU OBJECTMenuRid"~Object"    /*DISABLED*/
   SUBMENU OPTIONMenuRid"O~ption"    /*DISABLED*/
   SUBMENU VIEWMenuRid     "~View"      /*DISABLED*/
   DEFAULT_HELP_MENU
MENU OBJECTMenuRid           "~Object"
   ITEM OBJECT_MODIFY    "~Modify" CHECKED
   ITEM OBJECT_DESTROY     "~Destroy"  CHECKABLE
   ITEM OBJECT_ASSOCIATE"~Associate" CHECKABLE
   ITEM OBJECT_EXECUTE     "~Execute"  CHECKABLE
   ITEM OBJECT_SOCIALIZE"~Socialize" CHECKABLE
     SEPARATOR
     ITEM OBJECT_PARENT     "~Parent" CHECKED
     ITEM OBJECT_CHILD   "~Child"  CHECKABLE
MENU OPTIONMenuRid           "O~ption"
   ITEM OPTION_MOVE         "~Move"      CHECKABLE
   ITEM OPTION_AUTO    "~Auto"       CHECKED
     ITEM OPTION_RELATION    "~Relations" CHECKABLE
     ITEM OPTION_NODE_INFO "~Node Information" CHECKABLE
MENU VIEWMenuRid             "~View"
   ITEM VIEW_NONE       "~None"    CHECKABLE
   ITEM VIEW_BUTTON         "~Button"    CHECKED
   ITEM VIEW_GRAPH_RECTANGLE        "~Rectangle"
CHECKABLE

STRING OBJECT_PARENT    "Parent"
STRING OBJECT_CHILD       "Child"

STRING VIEW_NONE       "None"
STRING VIEW_BUTTON           "Button"
STRING VIEW_GRAPH_RECTANGLE "Rectangle"

STRING PROCESS_FUNCTION    "Function"
STRING PROCESS_EXEC          "Executable"
```

```
#ifdef WSWIN

/*Windows Resources:*/
#transparent my$
CURSOR_GRAPHER CURSOR "grapher.cur"
CURSOR_PARENTCHILD CURSOR "parnchld.cur"
my$
#endif

#ifdef WSMAC

/* Macintosh Resources: */
#transparent $$$
include "Grapher.rsrc";
$$$

#endif
```

```
/*****************************************************************
**********
 *
   Copyright(c) 1994, 1995 SSi.
```

CGrapherStartup.cxx

```
   File Name:    CGrapherStartup.cxx

   Modifications:              None
                          *

*****************************************************************
*********/

#ifdef DOS
#include "CGraphAp.h"
#else
#include "CGrapherApp.h"
#endif
#include "Grapher.h"
extern "C"
{

  int main(int argc, char** argv)
  {
    CGrapherApp theApplication;
    theApplication.Go(argc, argv, MENUBarRid, ABOUTBOXRid,
          "Grapher", "Grapher Application", "Grapher Task
Window");
    return 0;
  }

}
```

```
/*****************************************************************
*********
 *
```

CGrapherApp.h

```
   File Name:    CGrapherApp.h
   Author:     Sanjeev K. Guleria
   Date:     Tue Apr 4 18:08:20:54 IST 1994

   Class:   CGrapherApp
   Inheritance:     CApplication
   Helper(s):    CGraphDoc, CGraphWin

   Purpose: This class overrides the set of methods of
CApplication.
   It initializes the inherited application and opens the
Task Window
   So that user may open a previous Graph on start afresh.

   Usage: You may rename this class and add or override
methods as
     nessesary.

   Override:

   Modifications:
   Added the definition of SetUpMenus()
   on 28-02-94 by Sanjeev K. Guleria
   Added the definition of
     aJunkIdList
     AddJunk()
     RemoveJunk()
     GetNewId()
     DoClose()
   on 02-03-94 by Sanjeev K. Guleria
   Added the definition of
     ~CGrapherApp()
     itsGraphFileNameList;
     itsGrapherAppFilePointer;
     GetNumFiles()
     FileId()
   on 03-03-94 by Sanjeev K. Guleria
  Changed the definition of FileId
     BOOLEAN IsFileOpen(CString& aFullFileName, int &theId);
  on 04-03-94 by Sanjeev K. Guleria
```

```
*
**********************************************************************
********/

#ifndef CGrapherApp_H
#define CGrapherApp_H

#include "PwrDef.h"
#include CApplication_i
#include CUnits_i
#include CWindow_i
#include CStringList_i
#include CList_i
class CGrapherApp : public CApplication
{
  public:
    CGrapherApp(void);

   virtual void StartUp(void);
   virtual void DoCommand(long theCommand,void* theData);
   virtual BOOLEAN DoNew(void);
   virtual BOOLEAN DoOpen(void);
   virtual void SetUpMenus(void);

private:

   CStringList itsGraphFileNameList;
   int GetNewId();
   int GetNumFiles();
   BOOLEAN IsFileThere(CString& aFullFileName, int &theId);

};

#endif //CGrapherApp_H
```

```
/*************************************************************
**********
 *
```

CGrapherApp.cxx

```
   File Name:    CGrapherApp.cxx
   Author:     Sanjeev K. Guleria
   Date:     Tue Apr 4 18:08:20:54 IST 1994


   Modifications:
   Added the definition of
     aJunkIdList
     AddJunk()
     RemoveJunk()
     GetNewId()
   on 02-03-94 by Sanjeev K. Guleria
                          *

*************************************************************
*********/
#ifdef STDH
#include "stream.h"
#endif
#ifdef DOS
#include "CGraphAp.h"
#include "CGraphDc.h"
#include "fstream.h"
#else
#include "CGrapherApp.h"
#include "CGrapherDoc.h"
#endif
#include "Grapher.h"

//////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
Usage: This constructs the instance of CGrapherApp.
Work done:
*  Here the flag for making this application an MDI is set.
*  The Task Window is maximized.                    */
//////////////////////////////////////////////////////////////
//////////////////
CGrapherApp::CGrapherApp(void) : CApplication()
{
```

```
set_value(NULL_WIN, ATTR_WIN_MDI, TRUE);
set_value(NULL_WIN, ATTR_WIN_PM_TWIN_STARTUP_STYLE,
  WSF_ICONIZABLE | WSF_SIZE | WSF_CLOSE | WSF_MAXIMIZED);
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
Usage: This function is called when the application starts
up.
Work Done: Functions are called that
* first hard-code the metrics used in developing the
system.
* Set the flag so that unit conversion is based on a
comparison of the
development metrics with the execution metrics which are
computed at run time.*/
////////////////////////////////////////////////////////////
////////////////////
void CGrapherApp::StartUp(void)
{
CApplication::StartUp();
//This is the code to get DevelopmentMetrics so that they
could be hard coded
  //xvt_note("Development Metrics are :(%ld,%ld)[%ld,%ld]",
      //get_value(TASK_WIN, ATTR_SCREEN_WIDTH),
      //get_value(TASK_WIN, ATTR_SCREEN_HEIGHT),
      //get_value(TASK_WIN, ATTR_SCREEN_HRES),
      //get_value(TASK_WIN, ATTR_SCREEN_VRES));
SetUnits(new CUnits);
GetUnits()->SetDevelopmentMetrics(640,480,96,96);
GetUnits()->SetDynamicMapping(TRUE);
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
    Usage: This is CGrapherApp level DoCommand handler  */
////////////////////////////////////////////////////////////
///////////////
void CGrapherApp::DoCommand(long theCommand,void* theData)
{
  switch (theCommand/GRAPHER_BASE)
  {
  case FALSE:
/*Work Done:  Here theCommand can be Id of any document
represented
```

```
  by a number less than GRAPHER_BASE. The supplied Filename
is searched
  for its existance. If TRUE then the filename is nullified
*/
    {
    CString * aStringPtr;
    int intId;
    intId = (int)theCommand;
    aStringPtr = (CString *)theData ;
  if (IsFileThere(*aStringPtr, intId))
    {
    *aStringPtr = "";
    }
  else
    itsGraphFileNameList.Replace(*aStringPtr,
(int)theCommand);
  }
  break;
 default:
  CApplication::DoCommand(theCommand, theData);
  }
}
////////////////////////////////////////////////////////////
////////////////////
/******     '    Description            **********
    Usage: This is CGrapherApp level Overridden DoNew
    Work Done : In this function it is Checked whether
there exists
    a New CGraphDoc which was closed without doing any
work. If YES
    then that one CGraphDoc Object is used.      */
////////////////////////////////////////////////////////////
///////////////
BOOLEAN CGrapherApp::DoNew(void)
{
  CGraphDoc* aDoc;
  CString aFullFileName;
  int intNumberOfNewGraphs =
itsGraphFileNameList.InList(aFullFileName);
  CStringIterator doTo(&itsGraphFileNameList);
  CString * aName ;
  int theId = -1, anId;
  if(intNumberOfNewGraphs)
    {
    while (aName = (CString *)doTo.NextString())
```

```
        if (*aName == aFullFileName)
          {
          anId = doTo.GetCurrentPosition();
          aDoc = (CGraphDoc*) FindDocument(anId);
          CGraphWin* aWin;
          if (aDoc)
            aWin = (CGraphWin*)aDoc->FindWindow(anId);
          if (!aWin)
            {
            theId = anId;
            break;
            }
          }
        }
    if (theId == -1)
      {
      theId = GetNewId();
      aDoc = new CGraphDoc(this, theId);
      PwrAssert(aDoc, 311, "Creation of CGraphDoc failed");
      itsGraphFileNameList.Insert(aFullFileName, theId);
        }
     BOOLEAN aResult = aDoc->DoNew();
     aDoc->SetWindowTo(theId);
     return aResult;
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is CGrapherApp level Overridden DoOpen
     Work Done : In this function it is Checked whether
there exists
     a CGraphDoc Object with same file name. If YES then
that one
     CGraphDoc Object is used and its Window is taken to the
front.
     In case Window is closed then a CGraphWin Object is
created.*/
////////////////////////////////////////////////////////////////
///////////////
BOOLEAN CGrapherApp::DoOpen(void)
{
      CGraphDoc aTempDoc(this, 0);
      BOOLEAN aResult = aTempDoc.DoOpen();
      if (aResult)
     {
```

```
    int anId = 0;
    CString aFileName(aTempDoc.LocalFileName());
    aResult = IsFileThere(aFileName, anId);
    if (!anId)
      {
      anId = GetNewId();
      itsGraphFileNameList.Insert(aFileName, anId);
      }
    if (aResult)
      {
      CGraphDoc* aDoc = (CGraphDoc*) FindDocument(anId);
      CGraphWin* aWin;
      if (aDoc)
        aWin = (CGraphWin*)aDoc->FindWindow(anId);
      if (!aWin)
        {
        aDoc->SetWindowTo(anId);
        }
      aDoc->FrontWindow();
      }
    else
      {
      CGraphDoc* aDoc = new CGraphDoc(aTempDoc);
      PwrAssert(aDoc, 311, "Creation of CGraphDoc
failed");
      aDoc->SetWindowTo(anId);
      }
    }
    return aResult;
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
    Usage: This is CGrapherApp level SetUpMenus handler
    Work Done : Some of the menu items are set/unset.*/
////////////////////////////////////////////////////////////
///////////////
void CGrapherApp::SetUpMenus(void)
{
  win_menu_enable(TASK_WIN, M_FILE_NEW, TRUE);
  win_menu_enable(TASK_WIN, M_FILE_OPEN, TRUE);
  win_menu_enable(TASK_WIN, OBJECTMenuRid, FALSE);
  win_menu_enable(TASK_WIN, OPTIONMenuRid, FALSE);
  win_menu_enable(TASK_WIN, VIEWMenuRid, FALSE);
}
```

```
////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
     Usage: This is CGrapherApp Local function.
     Work Done : Returns a NewId to be assigned to a
CGraphDoc Object */
////////////////////////////////////////////////////////////////
/////////////////
int CGrapherApp::GetNewId()
{
return (GetNumFiles()+1);
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
     Usage: This is CGrapherApp Local function.
     Work Done : Returns the Number of CgraphDoc Objects
Opened till
     that particular moment.                    */
////////////////////////////////////////////////////////////////
////////////////
int CGrapherApp::GetNumFiles()
{
return itsGraphFileNameList.NumItems();
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
     Usage: This is CGrapherApp Local function.
     Work Done : Checks if the supplied aFullFileName has
been assigned
     to a CGraphdoc Object till that particular moment.  */
////////////////////////////////////////////////////////////////
////////////////
BOOLEAN CGrapherApp::IsFileThere(CString& aFullFileName, int
&theId)
{
BOOLEAN aResult = FALSE;

CStringIterator doTo(&itsGraphFileNameList);
CString * aName ;
while (aName = (CString *)doTo.NextString())
  if (*aName == aFullFileName)
     {
     theId = doTo.GetCurrentPosition();
```

```
        aResult = TRUE;
        break;
        }
return (aResult);
}
```

```
/*****************************************************************
*********
 *
   Copyright (c) 1993, 1994 XVT Software, Inc.
```

CGraphDoc.h

```
   File Name:    CGraphDoc.h
   Author:     Sanjeev K. Guleria
   Date:     Fri Apr 22 18:25:38:32 IST 1994

   Class:   CGraphDoc
   Inheritance:     CDocument
   Helper(s):

   Purpose: This is the class derived from CDocument. It
   serves the purpose of Graph as a document.

   Usage:You may rename this class and add or override
       methods as nessesary.

   Override:
    BuildWindow(void);
    DoCommand(long theCommand,void* theData=NULL);
  DoOpen(void);

   Modifications:
   Added the definition of aGrapherApp
   on 03-03-94 by Sanjeev K. Guleria
CString CGraphDoc::LocalFileName(void)
   on 04-03-94 by Sanjeev K. Guleria
    CGraphDoc(const CGraphDoc& theGraphDoc);
   on 07-03-94 by Sanjeev K. Guleria
  void SetWindowTo(int theId);
   on 08-03-94 by Sanjeev K. Guleria
                    *

*****************************************************************
********/

#ifndef CGraphDoc_HXX
#define CGraphDoc_HXX

#include "PwrDef.h"
#include CDocument_i
#include CList_i
```

```
#include COrderedList_i
#include CStringList_i
class CGraphWin;
class CGraphNode;

class CGraphDoc : public CDocument
{
 public:

    CGraphDoc(CApplication *theApplication,int theId );
          //constructor
    CGraphDoc(const CGraphDoc& theGraphDoc);
          //copy constructor
  ~CGraphDoc();
          //destructor
    virtual void BuildWindow(void);
          //used to build the window
    virtual void DoCommand(long theCommand,void*
theData=NULL);
          //Over ridden DoCommand
  virtual BOOLEAN DoOpen(void);
          //Over ridden DoOpen
  virtual void UpdateMenus(void);
          //Over ridden UpdateMenus
  virtual BOOLEAN DoSave(void);
          //Over ridden DoSave
  virtual BOOLEAN DoSaveAs(void);
          //Over ridden DoSaveAs
  CString LocalFileName(void);
          //returns the full filename with path
  void SetWindowTo(int theId);
          //The Window and document is set to theId
  void FrontWindow(void);
          //The Window of this document is brought to focus
  CString GetInfoList(int theNodeId, int
theInfoListPosition);
          /*Information List of theNodeId corresponding to
          theInfoListPosition is returned */
  void SetInfoList(int theGraphNodeId, int
theInfoListPosition, CString theString);
          /*Information List of theNodeId corresponding to
          theInfoListPosition is replaced by theString*/
  CString GetInfoString(CString theString, int
theInfoListPosition);
          /*Information List in theString corresponding to
```

```
        theInfoListPosition is returned */
  CString SetInfoString(CString theString, int
theInfoListPosition, CString theNewString);
        /*Information List in theString corresponding to
        theInfoListPosition is replaced by theNewString*/
  int GetRelation(int theOtherGraphNodeId, int
theRelationType,
                        CString theRelationString);
  int GetRelationPosition(int theOtherGraphNodeId, int
theRelationType,
                        CString theRelationString);
  CString SetRelation(int theOtherGraphNodeId, int
theRelationType,
                        CString theRelationString);
  CString UnSetRelation(int theOtherGraphNodeId, int
theRelationType,
                        CString theRelationString);
  CString SetUpRelation(CString theString, int
theFirstRelationType,
                        int theOtherRelationType);
private:
  CStringList itsGraphNodeInfoList;
  CStringList itsRelationInfoList;
  CList itsJunkNodeIdList;
  CString itsGraphNodeRelation;
  /*a CString object of relations between the active graph
nodes.*/
              /*an ordered list of inactive GraphNodeId's*/
  int itsRelationSession;
    unsigned itsNodeInfoMode;

  void AddJunk(int theNodeId);
  BOOLEAN RemoveJunk(int * theNodeId);
  int GetNewId(void);
  int GetNumNodes(void);
  void RelationSessionProcess(int intId, int *intPtr);
  void NewObject(void);
  void DeleteObject(CString * theStringPtr);
  void UpdateView(CString * theStringPtr);
  void UpdateRelationInfo(CString * theStringPtr);
  void UpdateProcess(CString *theStringPtr);
  void GetOtherRelationId(CString *theStringPtr);

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

```
};

#endif //CGraphDoc_HXX
```

```
/*****************************************************************
**********
 *
```

CGraphDoc.cxx

```
   File Name:    CGraphDoc.cxx
   Author:     Sanjeev K. Guleria

   Modifications:
   Added the definition of
   on 03-03-94 by Sanjeev K. Guleria
                          *

*****************************************************************
*********/
#ifdef DOS
#include "CGraphAp.h"
#include "CGraphDc.h"
#include "CGraphWn.h"
#include "CGraphNd.h"
#include "fstream.h"
#else
#include "CGrapherApp.h"
#include "CGrapherDoc.h"
#include "CGrapherWin.h"
#include "CGrapherNode.h"
#include "stream.h"
#endif
#include "CUtility.h"
#include "Grapher.h"

//////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
Usage: This constructs the instance of CGraphDoc.
Work done:
* The attributes are initialized.                    */
//////////////////////////////////////////////////////////////////
//////////////////
CGraphDoc::CGraphDoc(CApplication *theApplication,int theId
) :
        CDocument(theApplication,theId)
{
itsRelationSession = FALSE;
itsNodeInfoMode = FALSE;
```

```
itsGraphNodeRelation = "0 ,0 ,0 ,0 ,";
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
Usage: This copy constructs the instance of CGraphDoc.
Work done:
* The attributes are initialized.                    */
////////////////////////////////////////////////////////////////
////////////////////
CGraphDoc::CGraphDoc(const CGraphDoc& theGraphDoc) :
          CDocument((CDocument&)theGraphDoc)
{
// copy constructor
itsRelationSession = FALSE;
itsNodeInfoMode = FALSE;
itsGraphNodeRelation = "0 ,0 ,0 ,0 ,";
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
Usage: This is called before destroying the instance of
CGraphDoc.
Work done:
* The memory is freed .                 */
////////////////////////////////////////////////////////////////
////////////////////
CGraphDoc::~CGraphDoc()
{
// destructor
  {
  CIterator doTo(itsJunkNodeIdList);
  int* aNum;
  while (aNum = (int* )doTo.Next())
    {
    delete aNum;
    }
  }
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
    Usage: This is CGraphDoc level DoCommand handler  */
////////////////////////////////////////////////////////////////
////////////////
```

```
void CGraphDoc::DoCommand(long theCommand,void* theData)
{
  switch (theCommand/GRAPHER_BASE)
  {
  case FALSE:
  if (itsRelationSession)
    RelationSessionProcess((int)theCommand, (int *)
theData);
  else
      {
      CGraphNode aGraphNode(this, (int)theCommand,
itsNodeInfoMode);
      aGraphNode.ExecuteProcess();
      }
  break;
  case TRUE:
  switch(theCommand)
    {
      case NEWObjectCmd:
        NewObject();
       break;
      case DELETEObjectCmd:
        DeleteObject((CString *)theData);
       break;
      case UPDATEViewCmd:
       UpdateView((CString *)theData);
       break;
      case NEWNodeRelationCmd:
          itsRelationSession = TRUE;
        break;
      case FLIPNodeRelationCmd:
        itsGraphNodeRelation =
SetUpRelation(itsGraphNodeRelation,
                          PARENT_RELATION, CHILD_RELATION);
        SetSave(TRUE);
        itsRelationSession = FALSE;
        break;
      case UPDATERelationInfoCmd:
       UpdateRelationInfo((CString *)theData);
       break;
      case UPDATEProcessCmd:
       UpdateProcess((CString *)theData);
       break;
      case PROCESS_FUNCTION:
       {
```

```
        CGraphWin * aWin = (CGraphWin *)FindWindow(GetId());
        if (aWin)
           {
           aWin->DoMenuCommand(OPTION_RELATION,FALSE,FALSE);
           }
        }
        break;
       case GETOtherRelationIdCmd:
        GetOtherRelationId((CString *)theData);
        break;
       case GETRelationNameCmd:
        {
        CString *aStringPtr;
        int intRelId;
        aStringPtr = (CString *)theData;
        intRelId = atoi(aStringPtr->GetCharPtr());
        *aStringPtr = itsRelationInfoList.Find(intRelId);
        *aStringPtr = GetInfoString(*aStringPtr,
RELATION_NAME);
        }
        break;
       case GETNodeTitleCmd:
        {
        CString *aStringPtr;
        int intNodeId;
        aStringPtr = (CString *)theData;
        intNodeId = atoi(aStringPtr->GetCharPtr());
        *aStringPtr = (((CGraphWin *)FindWindow(GetId()))->
              FindGraphNodeView(intNodeId))->GetTitle();
         }
         break;
     case SENDRelationCmd:
        {
        CGraphWin * aWin = (CGraphWin *)FindWindow(GetId());
        if (aWin)
           {
           CString *aStringPtr;
           int intFirstId, intOtherId;
           aStringPtr = (CString *)theData;
           intFirstId = atoi(aStringPtr->GetCharPtr());
           intOtherId = atoi(GetInfoString(*aStringPtr,
2).GetCharPtr());
           aWin->RelationDraw(intFirstId, intOtherId);
           }
        }
```

```
        break;
       case KILLRelationCmd:
        {
        CString *aStringPtr;
        int intFirstId, intOtherId, intRelationType;
        aStringPtr = (CString *)theData;
        intFirstId = atoi(aStringPtr->GetCharPtr());
        intOtherId = atoi(GetInfoString(*aStringPtr,
2).GetCharPtr());
        intRelationType = atoi(GetInfoString(*aStringPtr,
3).GetCharPtr());
        CGraphNode aTempGraphNode(this, intFirstId, 0);

   aTempGraphNode.UnSetRelation(intOtherId,CHILD_RELATION);

   aTempGraphNode.UnSetRelation(intOtherId,PARENT_RELATION);
        }
        break;
     case  VIEWRelationCmd:
        {
        CString *aStringPtr;
        int intId;
        aStringPtr = (CString *)theData;
        if (aStringPtr)
          {
          intId = atoi(aStringPtr->GetCharPtr());
          CGraphNode aTempGraphNode(this, intId, 0);
          aTempGraphNode.SendRelation(SENDRelationCmd,
CHILD_RELATION, 1);
          aTempGraphNode.SendRelation(SENDRelationCmd,
PARENT_RELATION, 0);
          }
        else
          {
          CStringIterator doTo(&itsGraphNodeInfoList);
          while (aStringPtr = (CString *)doTo.NextString())
            {
            intId = atoi(aStringPtr->GetCharPtr());
            CGraphNode aTempGraphNode(this, intId, 0);

   aTempGraphNode.SendRelation(SENDRelationCmd,CHILD_RELATIO
N, 1);
            }
          }
            break;
```

```
            }
        case SETNodeInfoModeCmd:
            {
            unsigned * uintPtr;
          uintPtr = ((unsigned *) theData);
          itsNodeInfoMode = *uintPtr;
          }
          break;
        case CLEARAllInfoCmd:
          {
          CStringIterator doRelTo(&itsRelationInfoList);
          CString *itsText;
         int intRelId;
         while(itsRelationInfoList.NumItems())
             {
            itsText = (CString *)doRelTo.NextString();
            intRelId = atoi(itsText->GetCharPtr());
            itsRelationInfoList.Remove(intRelId);
            }
          CStringIterator doTo(&itsGraphNodeInfoList);
         int intNodeId;
         while(GetNumNodes())
             {
            itsText = (CString *)doTo.NextString();
            intNodeId = atoi(itsText->GetCharPtr());
            itsGraphNodeInfoList.Remove(intNodeId);
            }
         }
         break;
        default:
        CDocument::DoCommand(theCommand,theData);
        break;
        }
      break;
  }
}
//////////////////////////////////////////////////////////////
//////////
/******          Description            **********
    Usage: This is used to update temporary work area for
         relation informationmake a new Object
    Work Done: theId is updated as a relation of one type
         or the other according to intPtr passed */
//////////////////////////////////////////////////////////////
//////////////
```

```
void CGraphDoc::RelationSessionProcess(int theId,int
*intPtr)
{
if (*intPtr)
   {
   BOOLEAN aCheck=FALSE;
   if (aCheck = GetRelation(theId, CHILD_RELATION,
itsGraphNodeRelation))
      {
      itsGraphNodeRelation = UnSetRelation(theId,
CHILD_RELATION, itsGraphNodeRelation);
       *intPtr = -1;
      }
   if (aCheck = GetRelation(theId, PARENT_RELATION,
itsGraphNodeRelation))
      itsGraphNodeRelation = UnSetRelation(theId,
PARENT_RELATION, itsGraphNodeRelation);
     else
      itsGraphNodeRelation = SetRelation(theId,
PARENT_RELATION, itsGraphNodeRelation);
     }
else
   {
   BOOLEAN aCheck=FALSE;
   if (aCheck = GetRelation(theId, PARENT_RELATION,
itsGraphNodeRelation))
      {
      itsGraphNodeRelation = UnSetRelation(theId,
PARENT_RELATION, itsGraphNodeRelation);
       *intPtr = -1;
      }
   if (aCheck = GetRelation(theId, CHILD_RELATION,
itsGraphNodeRelation))
      itsGraphNodeRelation = UnSetRelation(theId,
CHILD_RELATION, itsGraphNodeRelation);
   else
      itsGraphNodeRelation = SetRelation(theId,
CHILD_RELATION, itsGraphNodeRelation);
   }
}
////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
     Usage: This is used to make a new Object
     Work Done: The Id is allocated,
```

```
     View is created. Information about GraphNode is
initialized   */
////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::NewObject(void)
{
SetSave(TRUE);
CString anyString;
int anId = GetNewId();
char charArray[10];
itoa(anId, charArray, 10);
anyString = charArray;
anyString += " ,0 ,0 ,0, ";
anyString += ((CGraphWin *)FindWindow(GetId()))-
>NewView(anId);
itsGraphNodeInfoList.Replace(anyString, anId);
}
////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
     Usage: This is used to Delete an Object
     Work Done: The Object with Id as anId has to be deleted
     So message is sent to that Node to send its relations
     with other Objects. Information of GraphNode is removed
     The Id of Node Object deleted is kept as junk*/
////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::DeleteObject(CString * theStringPtr)
{
int anId;
anId = atoi(theStringPtr->GetCharPtr());

CGraphNode aTempGraphNode(this, anId, 0);
aTempGraphNode.SendRelation(KILLRelationCmd, CHILD_RELATION,
0);
aTempGraphNode.SendRelation(KILLRelationCmd,
PARENT_RELATION, 0);
itsGraphNodeInfoList.Remove(anId);
AddJunk(anId);
}
////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
     Usage: This is used to Update View Information
     Work Done: The View associated to a node is updated*/
```

```
////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::UpdateView(CString * theStringPtr)
{
SetSave(TRUE);
char* AnyCharPtr;
CUtility aUtility;
CString aString;
int anId;
AnyCharPtr = aUtility.GetNthList(theStringPtr->GetCharPtr(),
                ITEM_SEPARATOR, 2, NULL);
aString = AnyCharPtr;
anId = atoi(theStringPtr->GetCharPtr());
SetInfoList(anId, VIEW_INFO, aString);
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
     Usage: This is used to Update Relation Information
     Work Done: The Relation Information is updated*/
////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::UpdateRelationInfo(CString * theStringPtr)
{
int intRelId;
intRelId = atoi(theStringPtr->GetCharPtr());
itsRelationInfoList.Replace(*theStringPtr, intRelId);
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
     Usage: This is used to Update Process
     Work Done: The Process associated to a node is
updated*/
////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::UpdateProcess(CString *theStringPtr)
{
SetSave(TRUE);
char* AnyCharPtr;
CUtility aUtility;
CString aString;
int anId;
AnyCharPtr = aUtility.GetNthList(theStringPtr->GetCharPtr(),
                    ITEM_SEPARATOR, 2, NULL);
```

```
aString = AnyCharPtr;
anId = atoi(theStringPtr->GetCharPtr());
SetInfoList(anId, PROCESS, aString);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                **********
     Usage: This function is used to Get the Id of the Other
     Relation related to this one as intRelId
     Work Done: The Other Id is taken from
itsRelationInfoList*/
////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::GetOtherRelationId(CString *theStringPtr)
{
int intRelId;
intRelId = atoi(theStringPtr->GetCharPtr());
*theStringPtr = itsRelationInfoList.Find(intRelId);
*theStringPtr = GetInfoString(*theStringPtr,
OTHER_RELATION_ID);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                **********
     Usage: This is CGraphDoc level Overridden DoOpen
     Work Done: Overridden to get rid of the call to
BuildWindow()
           Check is made that file type is not any other
than
           specified*/
////////////////////////////////////////////////////////////////
///////////////
BOOLEAN CGraphDoc::DoOpen(void)
{
  if (!itsXVTFilePointer)
  {
    itsXVTFilePointer = new FILE_SPEC;
    strcpy(itsXVTFilePointer->type, GRAPHER_TYPE);//[0] =
NULL;
    itsXVTFilePointer->name[0] = NULL;
    get_default_dir(&itsXVTFilePointer->dir);
  }
  //Handle file menu open by bringing up the open file
dialog box.
```

```
  switch (open_file_dlg(itsXVTFilePointer, "Enter a new file
to open:"))
  {
    case FL_OK:
      if (strcmp(itsXVTFilePointer->type, GRAPHER_TYPE))
       {
       xvt_note("Cannot open a .%s file instead of a .%s,
So Open is Cancelled",
                      itsXVTFilePointer->type,
GRAPHER_TYPE);
       return FALSE;
       }
      chg_dir(&itsXVTFilePointer->dir);
      //BuildWindow();
      return TRUE;

    case FL_BAD:
    case FL_CANCEL:
      return FALSE;
  }

  return FALSE;
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description            **********
    Usage: This is CGraphDoc level Overridden BuildWindow

    Work Done : In this function an instance of CGraphWin
is
    made. After that it is checked whether there exists
    a file associated with this CGraphDoc Object. If TRUE
    then that file is read and the window Object
initialised.       */
////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::BuildWindow(void)
{
  // Build an instance of CGraphWin:
  int intGraphId = GetId();
  CGraphWin *aWin = new CGraphWin(this,
CRect(20+intGraphId,40+intGraphId,
              300 + intGraphId,300+intGraphId), "Graph
Window",
              WSF_CLOSE|WSF_SIZE|WSF_ICONIZABLE);
```

```
  aWin->SetId(intGraphId);
CString fileText;
int intNodeId;
if (itsXVTFilePointer)
   {
   ifstream fileStream(itsXVTFilePointer->name);
   if (fileStream)
      {
      fileStream >> fileText;
      int aRelationCount = atoi(fileText.GetCharPtr());
      for (int i=0;i < aRelationCount;i++)
      {
      fileText.Clear();
      fileStream >> fileText;
      int intRelationId;
      intRelationId = atoi(fileText.GetCharPtr());
      itsRelationInfoList.Insert(fileText, intRelationId);
        fileText = GetInfoString(fileText, RELATION_NAME);
      aWin->RelationName(intRelationId, fileText);
      }
      for (i=1;(!fileStream.eof());i++)
      {
      fileText.Clear();
      fileStream >> fileText;
      if (fileText.Length()>0)
         {
         intNodeId = atoi(fileText.GetCharPtr());
         while(i<intNodeId)
            {
            AddJunk(i);
            i++;
            }
         itsGraphNodeInfoList.Insert(fileText, intNodeId);
         fileText = GetInfoList(intNodeId, VIEW_INFO);
         aWin->NewView(intNodeId, fileText);
         }
      }
      }
   }
else
   if (GetNumNodes()>0)
   {
     CStringIterator doTo(&itsGraphNodeInfoList);
     CString *itsText;
     while (itsText = (CString *)doTo.NextString())
```

```
    {
    intNodeId = atoi(itsText->GetCharPtr());
    fileText = GetInfoList(intNodeId, VIEW_INFO);
    aWin->NewView(intNodeId, fileText);
    }
  }
  aWin->IGraphWin(TRUE, "Graph Window", TRUE);
}
////////////////////////////////////////////////////////////////
/////////////////
/******          Description              **********
    Usage: This is CGraphDoc level Overridden UpdateMenus

    Work Done : In this function the Menu Options for Save
    and SaveAs Options are set according to the status of
    this Object.       */
////////////////////////////////////////////////////////////////
/////////////
void CGraphDoc::UpdateMenus(void)
{
  CWindow* win;
  WINDOW XVT_Win;
  CIterator doTo(itsWindows);
  while (win = (CWindow*) doTo.Next())
  {
    XVT_Win = win->GetXVTWindow();
    if (XVT_Win)
    {
    win_menu_enable(XVT_Win, M_FILE_SAVE_AS, NeedsSaving());

    if (itsXVTFilePointer)
     {
          win_menu_enable(XVT_Win, M_FILE_SAVE,
NeedsSaving());
       }
    }
  }
}
////////////////////////////////////////////////////////////////
/////////////////
/******          Description              **********
    Usage: This is CGraphDoc level Overridden DoSave

    Work Done : In the data of CGrpahDoc Object is stored
    in a file.       */
```

```
//////////////////////////////////////////////////////////////////
///////////////
BOOLEAN CGraphDoc::DoSave(void)
{
  // Handle File menu Save

  if (!CDocument::DoSave())
    return FALSE;

  // Save the file:
  save_dir();
  chg_dir(&itsXVTFilePointer->dir);
  CString * itsText;
  int intRelationCount;
  {
    ofstream fileStream(itsXVTFilePointer->name);
  if ((intRelationCount = itsRelationInfoList.NumItems())>0)
  {
    CStringIterator doTo(&itsRelationInfoList);
    char charRelationCount[10];
    sprintf(charRelationCount,"%d",intRelationCount);
    CString aString = charRelationCount;
    fileStream << aString;
    fileStream << "\n";

    while (itsText = (CString *)doTo.NextString())
     {
     fileStream << *itsText;
     fileStream << "\n";
     }

  }
  if (GetNumNodes()>0)
  {
    CStringIterator doTo(&itsGraphNodeInfoList);
    while (itsText = (CString *)doTo.NextString())
     {
     fileStream << *itsText;
     fileStream << "\n";
     }
  }
    fileStream.close();
  }
  restore_dir();
  return TRUE;
```

```
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is CGraphDoc level Overridden DoSaveAs

     Work Done : In this function it is checked whether
     there exists a file associated with this CGraphDoc
Object.
     If TRUEthen that filename is stored otherwise a new
     File-specification structure is initialized.
     Check is made that file type is not any other than
specified.*/
////////////////////////////////////////////////////////////////
///////////////
BOOLEAN CGraphDoc::DoSaveAs(void)
{
CString anOldFileNameString;

  if (!itsXVTFilePointer)
   anOldFileNameString = LocalFileName();
  else
   {
    itsXVTFilePointer = new FILE_SPEC;
    strcpy(itsXVTFilePointer->type, GRAPHER_TYPE);
    itsXVTFilePointer->name[0] = NULL;
    get_default_dir(&itsXVTFilePointer->dir);
   }
  switch (save_file_dlg(itsXVTFilePointer, "Save as:"))
  {
    case FL_OK:
      {
      if (strcmp(itsXVTFilePointer->type, GRAPHER_TYPE))
       {
       xvt_note("Cannot open a .%s file instead of a .%s,
So Open is Cancelled",
                    itsXVTFilePointer->type,
GRAPHER_TYPE);
       return FALSE;
       }
      CString aString;
      aString = LocalFileName();
      if (anOldFileNameString == aString)
       return FALSE;
    CDocument::DoCommand(GetId(),&aString);
```

```
    if (aString.Length() > 0)
       {
       SetWindowTo(GetId());
         return DoSave();
         }
       else
          {
          xvt_note("Grapher cannot give a Graph the same
name as an open Graph");
          SetSave(TRUE);
          delete itsXVTFilePointer;
          itsXVTFilePointer = NULL;
          return FALSE;
          }
       }
    case FL_BAD:
    case FL_CANCEL:
      return FALSE;
  }

  return FALSE;
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
     Usage: This is used to Set Window To theId
     Work Done: If theId is other than the current Id of
this Graph
     Then Id is changed. If Window is not existing then
BuildWindow
     is called. Finally theId is added to the Title of the
Window*/
////////////////////////////////////////////////////////////////
//////////////
void CGraphDoc::SetWindowTo(int theId)
{
  char aFileName[SZ_FNAME+10];
  int anId = GetId();
  CGraphWin * aWin = (CGraphWin *)FindWindow(anId);
  if (anId != theId)
   {
   anId = theId;
   SetId(anId);
   if (aWin)
     aWin->SetId(anId);
```

```
    }
    if (!aWin)
    {
    BuildWindow();
    aWin = (CGraphWin *)FindWindow(anId);
    }
    aFileName[0] = '\0';
    sprintf(aFileName,"<%d>", anId);
    if (itsXVTFilePointer)
    strcat(aFileName,itsXVTFilePointer->name);
    else
    strcat(aFileName,aWin->GetTitle().GetCharPtr());
    aWin->SetTitle(aFileName);
}
////////////////////////////////////////////////////////////////////
////////////////////
/******          Description                **********
     Usage: This is used to set focus to the Window of this
Graph
     Work Done: Window is set to be at front*/
////////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::FrontWindow(void)
{
set_front_window((FindWindow(GetId()))->GetXVTWindow());
}

////////////////////////////////////////////////////////////////////
////////////////////
/******          Description                **********
     Usage: This is used to get File name associated to this
Graph
          in the Local environment
     Work Done: Name is built up from the file-specification
available*/
////////////////////////////////////////////////////////////////////
///////////////
CString CGraphDoc::LocalFileName(void)
{
char aPath[SZ_FNAME+1];
dir_to_str(&itsXVTFilePointer->dir, aPath, SZ_FNAME+1);
CString aFullFileName(aPath);
aFullFileName += "\\";
aFullFileName += itsXVTFilePointer->name;
return aFullFileName;
```

```
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to get the list of theGraphNodeId
     at theInfoListPosition
     Work Done:*/
////////////////////////////////////////////////////////////////
////////////////
CString CGraphDoc::GetInfoList(int theGraphNodeId, int
theInfoListPosition)
{
return
GetInfoString(itsGraphNodeInfoList.Find(theGraphNodeId),
                              theInfoListPosition);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to get the list in theString
     at theInfoListPosition
     Work Done: If theInfoListPosition is LAST_INFO then
remaining
            string is returned Otherwise that particular
            String is returned*/
////////////////////////////////////////////////////////////////
////////////////
CString CGraphDoc::GetInfoString(CString theString, int
theInfoListPosition)
{
char * aTransitionString;
CString anOutputString("");
int aCharCount = 0;
CUtility aUtility;
aTransitionString =
aUtility.GetNthList(theString.GetCharPtr(),
               LIST_SEPARATOR, theInfoListPosition,
&aCharCount);
if (theInfoListPosition == LAST_INFO)
  anOutputString = aTransitionString;
else
  anOutputString.Append(aTransitionString, aCharCount);
return(anOutputString);
}
```

```
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is used to set the list of theGraphNodeId
     at theInfoListPosition
     Work Done: If theInfoListPosition is LAST_INFO then
remaining
            string is returned Otherwise that particular
            String is returned*/
////////////////////////////////////////////////////////////////
///////////////
void CGraphDoc::SetInfoList(int theGraphNodeId, int
theInfoListPosition, CString theString)
{
CUtility aUtility;
CString aTransitionString;
if (theInfoListPosition == LAST_INFO)
{
aTransitionString =
itsGraphNodeInfoList.Find(theGraphNodeId);
aUtility.NullTerminateNthList(aTransitionString.GetCharPtr()
,
                    LIST_SEPARATOR, theInfoListPosition -
1);
aTransitionString = aTransitionString.GetCharPtr();
aTransitionString += LIST_SEPARATOR;
aTransitionString += theString;
}
else
aTransitionString =
SetInfoString(itsGraphNodeInfoList.Find(theGraphNodeId),
                    theInfoListPosition, theString);

itsGraphNodeInfoList.Replace(aTransitionString,
theGraphNodeId);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is used to set the list in theString at
theInfoListPosition
     Work Done: This is achieved by replacing theString at
theInfoListPosition by          theNewString*/
////////////////////////////////////////////////////////////////
///////////////
```

```
CString CGraphDoc::SetInfoString(CString theString,
            int theInfoListPosition, CString theNewString)
{
CString anOutputString;
int aCharCount = 0;
CUtility aUtility;

char * charPtrTemp;

int intLength = theString.Length() + theNewString.Length() +
1;
charPtrTemp = new char[intLength];

aUtility.ReplaceNthGivenList(theString.GetCharPtr(),
                LIST_SEPARATOR, theInfoListPosition,
                theNewString.GetCharPtr(), charPtrTemp);
anOutputString = charPtrTemp;
delete[] charPtrTemp;
return(anOutputString);
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is used to get new Id to be given to a node
     Work Done: Checked if itsJunkNodeIdList is Empty
            If TRUE then new Id is one plus the previous
            number of nodes Otherwise its taken from
            itsJunkNodeIdList*/
////////////////////////////////////////////////////////////
////////////////
int CGraphDoc::GetNewId()
{
int anId, *intPtr;
if (itsJunkNodeIdList.IsEmpty())
   {
   anId = (GetNumNodes()+1);
   }
else
   {
   CIterator doTo(itsJunkNodeIdList);
   intPtr = (int*)doTo.Next();
   anId = *intPtr;
   RemoveJunk(intPtr);
   }
return anId;
```

```
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to Get Number of Nodes in the Graph
     Work Done: */
////////////////////////////////////////////////////////////////
////////////////
int CGraphDoc::GetNumNodes()
{
return itsGraphNodeInfoList.NumItems();
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to Add Junk
     Work Done: */
////////////////////////////////////////////////////////////////
////////////////
void CGraphDoc::AddJunk(int theNodeId)
{
int * anId = new int;
*anId = theNodeId;
itsJunkNodeIdList.Insert((void*)anId);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to Remove Junk
     Work Done: */
////////////////////////////////////////////////////////////////
////////////////
BOOLEAN  CGraphDoc::RemoveJunk(int * theNodeId)
{
return itsJunkNodeIdList.Remove((void*)theNodeId);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description                ***********
     Usage: This is used to get the number of times
theOtherGraphNodeId
            is defined in the temporary work area
     Work Done: For this the appropriate list is extracted
            and the count is get from there.      */
```

```
////////////////////////////////////////////////////////////////
///////////////
int CGraphDoc::GetRelation(int theOtherGraphNodeId, int
theRelationType, CString theRelationString)
{
CString aString = GetInfoString(theRelationString,
theRelationType);
int intTotalRelations = atoi(aString.GetCharPtr());
char *charPtrTemp;
int intTemp, intNumChars, intNumRelations = 0;
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
   {
   charPtrTemp = aUtility.GetNthList(aString.GetCharPtr(),
ITEM_SEPARATOR,
                                aCount, &intNumChars);
   intTemp = atoi(charPtrTemp);
   if (intTemp == theOtherGraphNodeId)
      intNumRelations++;
   }
return (intNumRelations);
}
////////////////////////////////////////////////////////////////
///////////////////
/******          Description              **********
     Usage: This is used to get the relation position in the
            temporary work area
     Work Done: For this the appropriate list is extracted
            and the relation position is get from there.
            That position is returned*/
////////////////////////////////////////////////////////////////
///////////////
int CGraphDoc::GetRelationPosition(int theOtherGraphNodeId,
int theRelationType, CString theRelationString)
{
int intTotalRelations =
atoi(theRelationString.GetCharPtr());
char *charPtrTemp;
int intTemp, intNumChars, intRelationPosition = 0;
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
```

```
   {
   charPtrTemp =
aUtility.GetNthList(theRelationString.GetCharPtr(),
ITEM_SEPARATOR,
                                aCount, &intNumChars);
   intTemp = atoi(charPtrTemp);
   if (intTemp == theOtherGraphNodeId)
      {
      intRelationPosition = aCount;
      break;
      }
   }
return (intRelationPosition);
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description                ***********
     Usage: This is used to set the relation in the
temporary work area
     Work Done: For this the appropriate list is extracted
            and the relation is set there. The changed work
area
            is returned*/
////////////////////////////////////////////////////////////////
///////////////
CString CGraphDoc::SetRelation(int theOtherGraphNodeId, int
theRelationType, CString theRelationString)
{
CString aString = GetInfoString(theRelationString,
theRelationType);
int intTotalRelations = atoi(aString.GetCharPtr());
char charArray[20];
char * charPtrTemp;
CUtility aUtility;
sprintf(charArray, "%d %d", intTotalRelations+1,
theOtherGraphNodeId);

int intLength = aString.Length() + strlen(charArray) + 1;
charPtrTemp = new char[intLength];

aUtility.ReplaceNthGivenList(aString.GetCharPtr(),
ITEM_SEPARATOR,
                        1, charArray, charPtrTemp);
aString = charPtrTemp;
```

```
theRelationString = SetInfoString(theRelationString,
theRelationType, aString);
delete[] charPtrTemp;
return theRelationString;
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is used to unset the relation in the
temporary work area
     Work Done: For this the appropriate list is extracted
           and the relation is unset there. The changed
work area
           is returned*/
////////////////////////////////////////////////////////////////
///////////////
CString CGraphDoc::UnSetRelation(int theOtherGraphNodeId,
           int theRelationType, CString theRelationString)
{
CString aString = GetInfoString(theRelationString,
theRelationType);
int intTotalRelations = atoi(aString.GetCharPtr());
int intRelationPosition = 0;
char charArray[20];
char * charPtrTemp;
CUtility aUtility;
BOOLEAN aResult = FALSE;
if (intRelationPosition =
GetRelationPosition(theOtherGraphNodeId,
                        theRelationType, aString))
  {
  int intLength = aString.Length() + 1;
  charPtrTemp = new char[intLength];
  aUtility.ReplaceNthGivenList(aString.GetCharPtr(),
ITEM_SEPARATOR,
              intRelationPosition, "", charPtrTemp);
    aString = charPtrTemp;
  sprintf(charArray, "%d", intTotalRelations-1);
  aUtility.ReplaceNthGivenList(aString.GetCharPtr(),
ITEM_SEPARATOR,
              1, charArray, charPtrTemp);
    aString = charPtrTemp;
  delete[] charPtrTemp;
  aResult = TRUE;
  }
```

```
theRelationString = SetInfoString(theRelationString,
theRelationType, aString);
return(theRelationString);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description            **********
     Usage: This is used to set up relations from the
temporary
          work area to the actual nodes.
     Work Done: Both lists are extracted
            Corresponding to each entry a CGraphNode Object
is
            created temporarily and whole of the other list
is
            sent to it to form the relations.*/
////////////////////////////////////////////////////////////////
///////////////
CString CGraphDoc::SetUpRelation(CString theString, int
theFirstRelationType, int theOtherRelationType)
{
  CGraphWin * aWin = (CGraphWin *)FindWindow(GetId());
CString theFirstRelationString;
CString theOtherRelationString;

theFirstRelationString = GetInfoString(theString,
theFirstRelationType);
theOtherRelationString = GetInfoString(theString,
theOtherRelationType);

int intTempId;
char * charPtrTemp;
CUtility aUtility;
int intTotalRelations =
atoi(theFirstRelationString.GetCharPtr());

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
  {
  charPtrTemp =
aUtility.GetNthList(theFirstRelationString.GetCharPtr(),
              ITEM_SEPARATOR, aCount, NULL);
  intTempId = atoi(charPtrTemp);
  CGraphNode aTempGraphNode(this, intTempId, 0);
```

```
  aTempGraphNode.NewRelation(theOtherRelationString,
theOtherRelationType, (theOtherRelationType ==
CHILD_RELATION));
    aWin->UpdateRelationViews(intTempId);
    }

intTotalRelations =
atoi(theOtherRelationString.GetCharPtr());

for(aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
  {
  charPtrTemp =
aUtility.GetNthList(theOtherRelationString.GetCharPtr(),
              ITEM_SEPARATOR, aCount, NULL);
  intTempId = atoi(charPtrTemp);
  CGraphNode aTempGraphNode(this, intTempId, 0);
  aTempGraphNode.NewRelation(theFirstRelationString,
theFirstRelationType, (theFirstRelationType ==
CHILD_RELATION));
    aWin->UpdateRelationViews(intTempId);
    }
theString = SetInfoString(theString, theFirstRelationType,
"0 ");
theString = SetInfoString(theString, theOtherRelationType,
"0 ");
return (theString);
}
```

CGraphWin.h

```
/*****************************************************************
*********
 *
   File Name:           CGraphWin.h
   Author:     Sanjeev K. Guleria
   Date:     Fri Apr 13 10:27:54:32 IST 1994

   Class:               CGraphWin
   Inheritance:         CWindow->CGraphWin
   Helper(s):

   Purpose:       This takes care of the view level of Graph.

   Usage:         You may rename this class and add overriden
methods as
      nessesary.

   Override:

   Modifications:
   Added the definition of
    ~CGraphWin()
*
   on 02-03-94 by Sanjeev K. Guleria
    CGraphWin(CGraphWin& theGraphWin);
   on 07-03-94 by Sanjeev K. Guleria

*****************************************************************
********/

#ifndef CGraphWin_HXX
#define CGraphWin_HXX

#include "PwrDef.h"
#include CUnits_i
#include CStringList_i
#include "CWindow.h"

class CScroller;
class CSketchPad;

class CGraphWin : public CWindow
{
```

```
  public:

    CGraphWin(CDocument *theDocument,
        const CRect& theRegion,
        const CString&           = NULLString,
        long theWindowAttributes = WSF_NONE,
        WIN_TYPE theWindowType   = W_DOC,
        int theMenuBarId         = MENU_BAR_RID);
    BOOLEAN  IGraphWin(BOOLEAN isBackgroundDrawn = TRUE,
            const CString& theTitle   = NULLString,
            BOOLEAN isVisible         = TRUE);

    virtual void DoCommand(long theCommand,void*
theData=NULL);
  virtual void DoMenuCommand(MENU_TAG theMenuItem,BOOLEAN
theShiftKey, BOOLEAN theControlKey);
    virtual BOOLEAN Close(void);
    CView* BuildObject(void);
    void GetInfo(void);
  CString NewView(int theGraphNodeId, CString
theInputString = NULL);
  void InitRelationNames(void);
  void InitRelationViews(void);
  void InitRelationViews(int theGraphNodeId);
  void UpdateRelationViews(void);
  void UpdateRelationViews(int theGraphNodeId);
  void DeleteRelationViews(void);
  void DeleteRelationViews(int theGraphNodeId);
  void NewRelation(int theFirstRelationId, int
theOtherRelationId,
                int theFirstRelPos, int theOtherRelPos,
                CString theFirstRelString,
                CString theOtherRelString);
  void RelationName(int theRelationId, CString
theRelationString);
  void RelationDraw(int theFirstNodeId, int
theOtherNodeId);
  void SetVariables(CString theInputString);
  CString GetVariables(void);
  void ReplaceVariables(CString& theInputString, int
theNodeViewItem);
  CView * FindGraphNodeView(int theNodeViewId);
private:
    int itsCurrentId;
       /*to store the Id of current view being handled.*/
```

```
    CRect itsRect;
       /* to store the Rectangle coordinates of current
view being
          handled*/
    CString itsName;
       /*to store the title of the current view being
handled*/
    CStringList itsRelationNameList;
       /*a list to store the names of relations as
         defined by user*/
    CScroller* itsScroller;
       /*the Scroller attached to the CGraphWin Object*/
    CSketchPad* itsSketchPad;
       /*the SketchPad attached to the CGraphWin Object*/
    unsigned itsMenuObject;
       /*to store the information about the currently
         selected Object Menu Item*/
    unsigned itsMenuObjectView;
       /*to store the information about the currently
         selected View Menu Item*/
  unsigned itsRelationsVisible;
       /*to store the information as to whether the
         relations are visible currently*/
  unsigned itsDraggable;
       /* to store the information as to whether the
         Objects are movable currently*/
    unsigned itsAutoSizable;
       /*to store the information as to whether the Objects
         are auto Sizable currently*/
    unsigned itsMovedOrSized;
       /* to store the information as to whether any of the
         Objects was moved currently*/
    unsigned itsMenuObjectSocialize;
       /*to store the information as to which relation
         could be set in the Socialized Mode*/
    unsigned itsNodeInfoMode;
       /* to store the information as to whether the
         information about Objects is to be shown or not*/
  void SetRectSize(CRect & theRect, CString theString);
};

#endif //CGraphWin_HXX
```

```
/*****************************************************************
**********
 *
   Copyright(c) 1993, 1994 SSi, Inc.
```

CGraphWin.cxx

```
   File Name:           CGraphWin.cxx
                        CGraphWn.cpp  (DOS)
   Author:     Sanjeev K. Guleria
   Date:    Fri Apr 13 13:56:26:32 IST 1994
   Modifications:

*

*****************************************************************
*********/
#include "windows.h"
#include "PwrDef.h"
#include CScroller_i
#include CSketchPad_i
//#include CRectangle_i
#include CLine_i
#include CPoint_i
#include NButton_i
#include CListBox_i
#ifdef DOS
#include "CAskDlg.h"
#include "CRelDlg.h"
#include "CRelLine.h"
#include "CGrphRct.h"
#else
//#include "CGraphWin.h"
//#include "CGraphDoc.h"
#include "CAskDialog.h"
#endif
#include "CGraphWn.h"
#include "CGraphDc.h"
#include "Grapher.h"
#include "CUtility.h"
////////////////////////////////////////////////////////////////
////////////////////
/******          Description            **********
Usage: This constructs the instance of CGraphWin.
Work done:
* The attributes are initialized.                    */
```

```
//////////////////////////////////////////////////////////////////
//////////////////
CGraphWin::CGraphWin(CDocument *theDocument,
          const CRect& theRegion,
          const CString& theTitle,
          long theAttributes,
          WIN_TYPE theWindowType,
          int theMenuBar)

: CWindow(theDocument, theRegion, theTitle,
    theAttributes, theWindowType, theMenuBar)
{
  // Add any code to create sub-objects here.
  itsMenuObjectView = VIEW_BUTTON;
  itsMenuObject = OBJECT_MODIFY;
  itsMenuObjectSocialize = OBJECT_PARENT;
  itsRelationsVisible = FALSE;
  itsDraggable = FALSE;
  itsAutoSizable = TRUE;
  itsNodeInfoMode = FALSE;

  // Setup the appripriate menus:
  win_menu_enable(GetXVTWindow(), M_FILE_OPEN, TRUE);
  win_menu_enable(GetXVTWindow(), M_FILE_NEW, TRUE);
  win_menu_enable(GetXVTWindow(), M_FILE_CLOSE, TRUE);

  win_menu_enable(GetXVTWindow(), OBJECT_PARENT, FALSE);
  win_menu_enable(GetXVTWindow(), OBJECT_CHILD, FALSE);
  win_menu_enable(GetXVTWindow(), OBJECTMenuRid,
!itsDraggable);
  win_menu_enable(GetXVTWindow(), OPTIONMenuRid, TRUE);
  win_menu_enable(GetXVTWindow(), VIEWMenuRid, TRUE);
  win_menu_check(GetXVTWindow(), OPTION_MOVE, itsDraggable);
  win_menu_check(GetXVTWindow(), OPTION_RELATION,
itsRelationsVisible);
  //set_cursor(GetXVTWindow(), CURSOR_GRAPHER);
  // Create the scroller:
  itsScroller = new CScroller(this, 1000, 1000);
  itsScroller->IScroller(TRUE, TRUE, TRUE, 10, 50);
  itsScroller->SetGlue(ALLSTICKY);
    // Create a sketchpad inside the scroller:
  itsSketchPad = new CSketchPad(itsScroller, itsScroller-
>GetVirtualFrame());
  itsSketchPad->SetCommand(SKETCHcmd);
  itsSketchPad->SetSketchEverywhere(FALSE);
```

```
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
Usage: This initializes the instance of CGraphWin.
Work done:
* The Relation Views are initialized.                    */
////////////////////////////////////////////////////////////////
/////////////////
BOOLEAN CGraphWin::IGraphWin(BOOLEAN isBackgroundDrawn,
            const CString& theTitle,
            BOOLEAN isVisible)
{
  InitRelationViews();
  return
CWindow::IWindow(isBackgroundDrawn,theTitle,isVisible);
}
////////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
Usage: This is overridden Close of CWindow.
Work done:
     Close window after giving the user a chance
     to update if in Socialize Mode
     to initialize Relation Names if not needed as yet
     to save if needed.
*                    */
////////////////////////////////////////////////////////////////
/////////////
BOOLEAN CGraphWin::Close(void)
{

  if (theMenuObject == OBJECT_SOCIALIZE)
  {
      switch(xvt_ask("YES","NO",NULL,
        "You are in Socialize Mode, Update before
Saving?"))
     {
     case RESP_DEFAULT:
       DoMenuCommand(OBJECT_EXECUTE,FALSE,FALSE);
       break;
     case RESP_2:
           break;
     }
   }
```

```
  if (itsDocument->NeedsSaving())
  {
   if (itsRelationNameList.Find(OBJECT_PARENT)==NULLString)
     InitRelationNames();
    ASK_RESPONSE response;
    do
    {
      response = xvt_ask("YES","NO","CANCEL",
        "[%s] Graph has changed, save before closing?",
        GetTitle().GetCharPtr());
    }
    while (response==RESP_DEFAULT && !itsDocument-
>DoSave()); // Save

    if (response == RESP_2)
// Don't save
      {
      itsDocument->SetSave(FALSE);
      }
    else

    if (response == RESP_3)
// Cancel
      return FALSE;
  }
    DoCommand(CLEARAllInfoCmd);
   return CWindow::Close();
}
////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
     Usage: This is CGraphWin level DoCommand handler  */
////////////////////////////////////////////////////////////
//////////////
void CGraphWin::DoCommand(long theCommand,void* theData)
{
  switch (theCommand/GRAPHER_BASE)
  {
  case FALSE:
   itsCurrentId = (int)theCommand;
   if (!itsDraggable)
    switch (itsMenuObject)
    {
  case OBJECT_MODIFY:
     {
```

```
    CView * item;
    int intId = (int)theCommand;
    item = FindGraphNodeView(intId);
    if (item)
      {
      char charArray[10];
      itsName = item->GetTitle();
      itsName = AskDlg(&itsName);
      item->SetTitle(itsName);
      CString aString(((CGraphDoc*)itsDocument)->
                  GetInfoList(intId, VIEW_INFO));
      itsRect = item->GetFrame();
      SetRectSize(itsRect, itsName);
      item->Size(itsRect);
      item->Draw();
      ReplaceVariables(aString, NODE_VIEW_RECT);
      ReplaceVariables(aString, NODE_VIEW_NAME);
      sprintf(charArray, "%d ", intId);
      aString = charArray + aString;
      CWindow::DoCommand(UPDATEViewCmd, &aString);
        UpdateRelationViews((int)theCommand);
      }
    }
    break;
  case OBJECT_DESTROY:
    {
    CView * item;
    int intId = (int)theCommand;
    item = FindGraphNodeView(intId);
    if (item)
      {
      char charArray[10];
      CString aString;
      sprintf(charArray, "%d ", intId);
      aString = charArray + aString;
      CWindow::DoCommand(DELETEObjectCmd, &aString);
      delete item;
      DeleteRelationViews(intId);
      }
    }
    break;
 case OBJECT_EXECUTE:
       CWindow::DoCommand(theCommand, theData);
        UpdateRelationViews((int)theCommand);
       break;
```

```
   case OBJECT_SOCIALIZE:
        {
        int aCheck;
          aCheck = (itsMenuObjectSocialize ==
OBJECT_PARENT);
        if
(itsRelationNameList.Find(OBJECT_PARENT)==NULLString)
          InitRelationNames();
        int intId;
        intId = (int)(theCommand);
        CWindow::DoCommand(theCommand, &aCheck);
        CRect aViewRect = itsSketchPad->FindSubview(intId)-
>GetFrame();
        CText *aText;
        CString aCurrentString =
itsRelationNameList.Find(itsMenuObjectSocialize);
        aText = (CText *)FindGraphNodeView(intId +
(GRAPHER_BASE/2));
        if (!aText)
          {
          aText = new CText(itsScroller,
CPoint(aViewRect.Left(), aViewRect.Top()));
          aText->SetOrigin(CPoint(0, - aText->GetHeight()));
          aText->SetId(intId + (GRAPHER_BASE/2));
          aText->SetCommand(RELATION_SHOW);
          }

        if ((aCurrentString == aText->GetText()) &&
            (aText->GetText().Length() > 0))
          {
          aText->SetText("");
          }
        else
          {
          aText->SetText(aCurrentString);
          }
        itsScroller->DoDraw(itsScroller->GetClippedFrame());
        }
        break;
      case OBJECT_ASSOCIATE:
          {
          CRect aListBoxRect =
FindGraphNodeView((int)theCommand)->GetFrame();
          CListBox *aListBoxPtr;
```

```
          aListBoxPtr = (CListBox
*)FindGraphNodeView(LISTBOXId);
          if (itsScroller->GetClippedFrame().IsPointInRect(
                CPoint(aListBoxRect.Left()-aListBoxPtr-
>GetFrame().Width() - 1,
                aListBoxRect.Top())))
             {
             aListBoxRect.Left(aListBoxRect.Left()-1-
aListBoxPtr->GetFrame().Width());
             }
          else
             {
             aListBoxRect.Left(aListBoxRect.Right()+1);
             }
          aListBoxRect.Width(aListBoxPtr-
>GetFrame().Width());
          aListBoxRect.Height(aListBoxPtr-
>GetFrame().Height());
          aListBoxPtr->Enable();
          aListBoxPtr->Size(aListBoxRect);
          aListBoxPtr->DoShow();
        CString aString(((CGraphDoc*)itsDocument)-
>GetInfoList((int)theCommand, PROCESS));
          int intProcessId = atoi(aString.GetCharPtr());
          if ((intProcessId) &&
(intProcessId>=PROCESS_FUNCTION)&&
                          (intProcessId<=PROCESS_EXEC))
             aListBoxPtr->SelectLine(intProcessId -
(PROCESS_FUNCTION));
          itsScroller->DoDraw();
          if ((itsNodeInfoMode)&&(intProcessId))
             xvt_note("Process Information [%s]",
aString.GetCharPtr());
          }
          break;
       }
   break;
   case TRUE:
     switch (theCommand)
     {
     case SKETCHcmd:
       if ((!itsDraggable)&&(itsMenuObject ==
OBJECT_MODIFY))
          GetInfo();
       else
```

```
        {
        CRect aSketchedRect = itsSketchPad-
>GetSketchedRegion();
        CRect aViewRect;
          CView* item;
          CIterator doTo(itsSketchPad->GetSubviews());
          SetSelectedView(NULL);
          while (item = (CView*) doTo.Next())
            {
          aViewRect = item->GetFrame();
          if ((aSketchedRect - aViewRect) == aViewRect)
            {
            DoCommand(item->GetId());
            }
            }
        }
       break;
      case FLIPNodeRelationCmd:
       itsSketchPad->DoDraw();
         CWindow::DoCommand(theCommand,theData);
       break;
      case WFSizeCmd:
       if (itsMovedOrSized)
            {
            CView* item;
            int intId;
            char charArray[10];
          if (item = GetSelectedView())
            {
              CIterator doSelectedTo(itsSketchPad-
>GetSelectedViews());
              SetSelectedView(NULL);
              while(item = (CView *)doSelectedTo.Next())
                {
                intId = item->GetId();
              CString aString(((CGraphDoc*)itsDocument)-
>GetInfoList(intId, VIEW_INFO));
              itsRect = item->GetFrame();
              ReplaceVariables(aString, NODE_VIEW_RECT);
              sprintf(charArray, "%d ", intId);
              aString = charArray + aString;
              CWindow::DoCommand(UPDATEViewCmd, &aString);
              UpdateRelationViews(intId);
              }
            }
```

```
            }
            break;
        case WFSelectCmd:
         itsMovedOrSized = TRUE;
            break;
        case WFDeselectCmd:
         itsMovedOrSized = FALSE;
            break;
        case LISTBOXSingleClickCmd:
            {
            int intSelectedIndex;
            CListBox *aListBoxPtr;
            aListBoxPtr = (CListBox
*)FindGraphNodeView(LISTBOXId);
            intSelectedIndex = aListBoxPtr->GetSelectedLine();
            aListBoxPtr->DoHide();
            aListBoxPtr->DoDraw();
            aListBoxPtr->Disable();
            DoCommand(intSelectedIndex + (PROCESS_FUNCTION));
         }
         break;
        case PROCESS_FUNCTION:
         {
         char charArray[20];
         xvt_note("Associate %s",
CString((int)theCommand).GetCharPtr());
         sprintf(charArray, "%d %d %d ", itsCurrentId,
                PROCESS_FUNCTION, PROCESS_FUNCTION);
             CString aString(charArray);
         CWindow::DoCommand(UPDATEProcessCmd,&aString);
         }
         break;
        case PROCESS_EXEC:
         {
         char charArray[10];
         xvt_note("Associate %s",
CString((int)theCommand).GetCharPtr());
         FILE_SPEC * anXVTFilePointer;
         anXVTFilePointer = new FILE_SPEC;
         strcpy(anXVTFilePointer->type, "EXE");
         anXVTFilePointer->name[0] = NULL;
         save_dir();
         get_default_dir(&anXVTFilePointer->dir);
         switch (open_file_dlg(anXVTFilePointer,
                  "Enter an executable file to Run:"))
```

```
          {
          case FL_OK:
             {
             chg_dir(&anXVTFilePointer->dir);
             char aPath[SZ_FNAME+1];
             dir_to_str(&anXVTFilePointer->dir, aPath,
SZ_FNAME+1);
             CString aFullFileName(aPath);
             aFullFileName += "\\";
             aFullFileName += anXVTFilePointer->name;
             sprintf(charArray, "%d %d ", itsCurrentId,
PROCESS_EXEC);
             aFullFileName = charArray + aFullFileName;

CWindow::DoCommand(UPDATEProcessCmd,&aFullFileName);
                }
          case FL_BAD:
          case FL_CANCEL:
            break;
          }
          restore_dir();
        }
        break;
       default:
         CWindow::DoCommand(theCommand,theData);
     }
   }
}
////////////////////////////////////////////////////////////
////////////////////
/******          Description              **********
     Usage: This is CGraphDoc level DoMenuCommand handler*/
////////////////////////////////////////////////////////////
////////////////
void CGraphWin::DoMenuCommand(MENU_TAG theMenuItem, BOOLEAN
theShiftKey,
                              BOOLEAN theControlKey)
{
  switch (theMenuItem)
  {
  case M_FILE_SAVE_AS:
     if
(itsRelationNameList.Find(OBJECT_PARENT)==NULLString)
       InitRelationNames();
```

```
  CWindow::DoMenuCommand(theMenuItem,theShiftKey,theControl
Key);
        break;
   case VIEW_NONE:
   case VIEW_BUTTON:
   case VIEW_GRAPH_RECTANGLE:
     win_menu_check(GetXVTWindow(), itsMenuObjectView,
FALSE);
     win_menu_check(GetXVTWindow(), theMenuItem, TRUE);
     itsMenuObjectView = theMenuItem;
     break;
   case OBJECT_MODIFY:
   case OBJECT_DESTROY:
   case OBJECT_ASSOCIATE:
   case OBJECT_EXECUTE :
   case OBJECT_SOCIALIZE :
     {
     win_menu_check(GetXVTWindow(), itsMenuObject, FALSE);
     win_menu_check(GetXVTWindow(), theMenuItem, TRUE);
     if (theMenuItem == OBJECT_SOCIALIZE)
         {
         CWindow::DoCommand(NEWNodeRelationCmd);
       win_menu_enable(GetXVTWindow(), OBJECT_PARENT,
TRUE);
       win_menu_enable(GetXVTWindow(), OBJECT_CHILD, TRUE);
       win_menu_enable(GetXVTWindow(), OPTION_MOVE, FALSE);
       //set_cursor(GetXVTWindow(), CURSOR_PARENTCHILD);
       }
       else
       {
       CView* item;
       CIterator doTo(itsScroller->GetSubviews());
       while (item = (CView*) doTo.Next())
          if (item->GetCommand()==RELATION_SHOW)
            delete (CText *)item;
       DoCommand(FLIPNodeRelationCmd);
       win_menu_enable(GetXVTWindow(), OBJECT_PARENT,
FALSE);
       win_menu_enable(GetXVTWindow(), OBJECT_CHILD,
FALSE);
       win_menu_enable(GetXVTWindow(), OPTION_MOVE, TRUE);
       //set_cursor(GetXVTWindow(), CURSOR_GRAPHER);
       }
       win_menu_enable(GetXVTWindow(), OPTION_AUTO,
```

```
                    (theMenuItem == OBJECT_MODIFY));
        if (theMenuItem==OBJECT_ASSOCIATE)
        {
          CString aLongestString;
          CString aStringArray[(PROCESS_EXEC) -
(PROCESS_FUNCTION) + 1];
          COrderedList anOrderedList;
          int intIndex, intId;
          for (intIndex=0,intId=(PROCESS_FUNCTION);
          (intId<=(PROCESS_EXEC));
          intIndex++,intId++)
               {
               aStringArray[intIndex] = CString(intId);

  anOrderedList.Insert(&aStringArray[intIndex],intIndex);
               if (aStringArray[intIndex].Length() >
                         aLongestString.Length())
                 {
                 aLongestString = aStringArray[intIndex];
                 }
               }
          CText aText(itsScroller, CPoint(0,0),
aLongestString);
          aText.Hide();
          CRect aListBoxRect;
          aListBoxRect.Width(aText.GetFrame().Width() + 8);
          aListBoxRect.Height(anOrderedList.NumItems() *
                              aText.GetFrame().Height());
          CListBox * aListBox = new CListBox(itsScroller,
aListBoxRect);
          aListBox-
>IListBox(&anOrderedList,FALSE,FALSE,FALSE,

  NULLcmd,LISTBOXSingleClickCmd,FALSE);
          aListBox->SetId(LISTBOXId);
        //set_cursor(GetXVTWindow(), CURSOR_ARROW);
         }
         else
         {
          CListBox *aListBoxPtr;
          aListBoxPtr = (CListBox
*)FindGraphNodeView(LISTBOXId);
          delete aListBoxPtr;
         }
     itsMenuObject = theMenuItem;
```

```
    }
    break;
  case OBJECT_PARENT:
  case OBJECT_CHILD:
    win_menu_check(GetXVTWindow(), itsMenuObjectSocialize,
FALSE);
    win_menu_check(GetXVTWindow(), theMenuItem, TRUE);
    itsMenuObjectSocialize = theMenuItem;
    break;
   case OPTION_MOVE:
      {
      itsDraggable = !itsDraggable;
      win_menu_enable(GetXVTWindow(), OBJECTMenuRid,
!itsDraggable);
      CView* item;
      CIterator doTo(itsSketchPad->GetSubviews());
      if (item = GetSelectedView())
        {
        CIterator doSelectedTo(itsSketchPad-
>GetSelectedViews());
        SetSelectedView(NULL);
        while(item = (CView *)doSelectedTo.Next())
          {
          item->Draw();
          }
        }
      while (item = (CView*) doTo.Next())
        {
      item->SetDragging(itsDraggable);
        }
    win_menu_check(GetXVTWindow(), theMenuItem,
itsDraggable);
    itsMovedOrSized = FALSE;
      break;
      }
   case OPTION_AUTO:
      {
      itsAutoSizable = !itsAutoSizable;
    win_menu_check(GetXVTWindow(), theMenuItem,
itsAutoSizable);
      break;
      }
   case OPTION_RELATION:
    {
    itsRelationsVisible = !itsRelationsVisible;
```

```
    win_menu_check(GetXVTWindow(), theMenuItem,
itsRelationsVisible);
    UpdateRelationViews();
    break;
    }
   case OPTION_NODE_INFO:
      {
      itsNodeInfoMode = !itsNodeInfoMode;
      DoCommand(SETNodeInfoModeCmd, &itsNodeInfoMode);
    win_menu_check(GetXVTWindow(), theMenuItem,
itsNodeInfoMode);
      break;
      }
   default:

CWindow::DoMenuCommand(theMenuItem,theShiftKey,theControlKey
);
  }
}
////////////////////////////////////////////////////////////
///////////
/******          Description              **********
    Usage: This is used to get information about a node
         to be formed.
    Work Done: a dialog box is shown that gets the name*/
////////////////////////////////////////////////////////////
///////////////
void CGraphWin::GetInfo(void)
{

  // First get the information:

  switch (itsMenuObjectView)
  {
    case VIEW_BUTTON:
    case VIEW_GRAPH_RECTANGLE:
    {
    itsName = AskDlg();
       if (itsName.Length()>0)
          DoCommand(NEWObjectCmd, &itsName);
      break;
      }
  case VIEW_NONE:
    xvt_note("NONE View is not supported yet");
    break;
```

```
  }

}
//////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to build a new Object
     Work Done: the Rectangle is sized according to name
            and the view is instantiated and returned  */
//////////////////////////////////////////////////////////////////
///////////////
CView* CGraphWin::BuildObject(void)
{
  CView* newObject;

  switch (itsMenuObjectView)
  {
    case VIEW_BUTTON:
     {
        SetRectSize(itsRect, itsName);
       newObject = new NButton(itsSketchPad, itsRect,
itsName);
      break;
       }
    case VIEW_GRAPH_RECTANGLE:
     {
        SetRectSize(itsRect, itsName);
       newObject = new CGraphRectangle(itsSketchPad,
itsRect, itsName);
      newObject->Draw();
      break;
     }
    case VIEW_NONE:
     break;

  }
newObject->SetDragging(itsDraggable);

  return newObject;
}
//////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to create a new View using
theInputString
```

```
        or system attributes
    Work Done: using theInputString  or CGraphWin
attributes
        the information about view is returned  */
////////////////////////////////////////////////////////////
///////////////
CString CGraphWin::NewView(int theGraphNodeId, CString
theInputString)
{
  unsigned aMenuObjectView;
  unsigned anAutoSizable;
  CView * aGraphView;
  CString aName("");
  int intLength = theInputString.Length();
  if (intLength >0)
  {
  aMenuObjectView = itsMenuObjectView;
  anAutoSizable = itsAutoSizable;
  SetVariables(theInputString);
  aGraphView = (CView *)BuildObject();

  itsMenuObjectView = aMenuObjectView;
  itsAutoSizable = anAutoSizable;
  }
  else
  {
  itsRect = itsSketchPad->GetSketchedRegion();

  aGraphView = (CView *)BuildObject();

  aName = GetVariables();
  }
  if (aGraphView)
     {
     aGraphView->SetId(theGraphNodeId);
     aGraphView->SetCommand((long)theGraphNodeId);
     }
  return (aName);
}
////////////////////////////////////////////////////////////
///////////
/******          Description              **********
    Usage: This is used to Set Rectangle Size
    Work Done: theRect is sized according to name
```

```
            if name cannot fit in theRect then Size is
increased
            if size is very big and autoSizing needs to be
done
            then size is decreased  */
////////////////////////////////////////////////////////////
///////////////
voi
```

```
  char * charPtrTemp;
  charPtrTemp = theInputString.GetCharPtr();
  itsAutoSizable = FALSE;
  itsName = aUtility.GetAfterNthList(charPtrTemp,
                    LIST_SEPARATOR, NODE_VIEW_NAME-1);
  aUtility.NullTerminateUptoNthList(charPtrTemp,
                    LIST_SEPARATOR, NODE_VIEW_NAME);
  for (int intListPosition = 1;
       intListPosition < NODE_VIEW_NAME;
                          intListPosition++)
    {
    switch(intListPosition)
      {
      case NODE_VIEW_TYPE:
        itsMenuObjectView =
atoi(aUtility.GetNthNullTerminatedList(
                    charPtrTemp, NODE_VIEW_TYPE));
      break;
      case NODE_VIEW_RECT:
        {
        charPtrTemp = aUtility.GetNthNullTerminatedList(
                    charPtrTemp, NODE_VIEW_RECT);
        switch(itsMenuObjectView)
          {
          case VIEW_BUTTON:
          case VIEW_GRAPH_RECTANGLE:
          for (int intTemp=0; intTemp <4; intTemp+=2)
          {
          UNITSRectArray[intTemp] =
            aUnit-
>HPhysicalToLogical(atoi(aUtility.GetNthList
            (charPtrTemp, ITEM_SEPARATOR, intTemp+1,
NULL)));
          UNITSRectArray[intTemp+1] =
            aUnit-
>VPhysicalToLogical(atoi(aUtility.GetNthList
            (charPtrTemp, ITEM_SEPARATOR, intTemp+2,
NULL)));
          }
          itsRect.SetRect(UNITSRectArray[0],
                    UNITSRectArray[1],
                    UNITSRectArray[2],
                    UNITSRectArray[3]);
          break;
          }
```

```
        }
      }
    }
}
////////////////////////////////////////////////////////////////
///////////
/******          Description                 **********
    Usage: This is used to Get Variables according to
         theInputString
    Work Done: a string is returned in which the
         attributes about View are stored according
         to the structure defined for the purpose*/
////////////////////////////////////////////////////////////////
///////////////
CString CGraphWin::GetVariables(void)
{
  char charArray[10];
  CString theOutputString;
  CUnits * aUnit = GetUnits();
  UNITS UNITSRectArray[4];
  for (int intListPosition = 1;
     intListPosition < NODE_VIEW_NAME;
     intListPosition++)
     {
     switch(intListPosition)
       {
       case NODE_VIEW_TYPE:
       {
       itoa(itsMenuObjectView, charArray, 10);
       theOutputString = charArray;
       break;
       }
       case NODE_VIEW_RECT:
       {
         switch(itsMenuObjectView)
         {
         case VIEW_GRAPH_RECTANGLE:
         case VIEW_BUTTON:
         UNITSRectArray[0] = itsRect.Left();
         UNITSRectArray[1] = itsRect.Top();
         UNITSRectArray[2] = itsRect.Right();
         UNITSRectArray[3] = itsRect.Bottom();
           for (int intTemp=0; intTemp < 4; intTemp+=2)
            {
```

```
            itoa(aUnit-
>HLogicalToPhysical(UNITSRectArray[intTemp]) ,
                        charArray, 10);
            theOutputString += charArray;
            theOutputString += ITEM_SEPARATOR;
            itoa(aUnit-
>VLogicalToPhysical(UNITSRectArray[intTemp+1]) ,
                        charArray, 10);
            theOutputString += charArray;
            theOutputString += ITEM_SEPARATOR;
            }
          break;
          }
        }
        }
     theOutputString += LIST_SEPARATOR;
     }
  theOutputString += itsName;
return theOutputString;
}
////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to Replace Variables in theString
          at theNodeViewItem position
     Work Done: In theString the attributes about View
          are replaced at theNodeViewItem position
          in the structure defined for the purpose*/
////////////////////////////////////////////////////////////
///////////////
void CGraphWin::ReplaceVariables(CString& theString, int
theNodeViewItem)
{
  CUtility aUtility;
  char charArray[10];
  char * charPtrTemp;
  CString theOutputString;
  CUnits * aUnit = GetUnits();
  UNITS UNITSRectArray[4];
  charPtrTemp = new char [2 * theString.Length()];  //(char
*)malloc(2 * theString.Length());
  switch(theNodeViewItem)
     {
     case NODE_VIEW_TYPE:
        {
```

```
        itoa(itsMenuObjectView, charArray, 10);
        aUtility.ReplaceNthGivenList(theString.GetCharPtr(),
          LIST_SEPARATOR, theNodeViewItem, charArray,
charPtrTemp);
        break;
        }
      case NODE_VIEW_RECT:
        {
        switch(itsMenuObjectView)
          {
          case VIEW_BUTTON:
          case VIEW_GRAPH_RECTANGLE:
          UNITSRectArray[0] = itsRect.Left();
          UNITSRectArray[1] = itsRect.Top();
          UNITSRectArray[2] = itsRect.Right();
          UNITSRectArray[3] = itsRect.Bottom();
            for (int intTemp=0; intTemp < 4; intTemp+=2)
             {
             itoa(aUnit-
>HLogicalToPhysical(UNITSRectArray[intTemp]) ,
                      charArray, 10);
             theOutputString += charArray;
             theOutputString += ITEM_SEPARATOR;
             itoa(aUnit-
>VLogicalToPhysical(UNITSRectArray[intTemp+1]) ,
                      charArray, 10);
             theOutputString += charArray;
             theOutputString += ITEM_SEPARATOR;
             }
          aUtility.ReplaceNthGivenList(theString.
                GetCharPtr(), LIST_SEPARATOR,
theNodeViewItem,
                theOutputString.GetCharPtr(), charPtrTemp);
          theString = charPtrTemp;
          break;
          }
        }
      break;
      case NODE_VIEW_NAME:
        {
        CUtility aUtility;
        aUtility.NullTerminateNthList(
          theString.GetCharPtr(), LIST_SEPARATOR,
(theNodeViewItem-1));
        strcpy(charPtrTemp, theString.GetCharPtr());
```

```
        theString = charPtrTemp;
        theString += LIST_SEPARATOR;
        theString += itsName;
        }
    break;
    }
delete[] charPtrTemp;
}
////////////////////////////////////////////////////////////
///////////
/******          Description                ***********
    Usage: This is used to Find Graph Node View with Id
         as theNodeViewId
    Work Done: View is returned by searching form
itsScroller */
////////////////////////////////////////////////////////////
///////////////
CView * CGraphWin::FindGraphNodeView(int theNodeViewId)
{
return itsScroller->FindSubview(theNodeViewId);
}
////////////////////////////////////////////////////////////
///////////
/******          Description                ***********
    Usage: This is used to Initialize Relation Names
    Work Done: Names of relations are got
           from the user and initialized  */
////////////////////////////////////////////////////////////
///////////////
void CGraphWin::InitRelationNames()
{
CString aFirstString;
CString anOtherString;
aFirstString = (CString)(OBJECT_PARENT);
anOtherString = (CString)(OBJECT_CHILD);
RelDlg(&aFirstString, &anOtherString);
NewRelation(OBJECT_PARENT, OBJECT_CHILD, PARENT_RELATION,
           CHILD_RELATION, aFirstString, anOtherString);
}
////////////////////////////////////////////////////////////
///////////
/******          Description                ***********
    Usage: This is used to Initialize Relation Views
    Work Done: Views are searched from itsScroller
           All Views of type RELATION_VIEW_ID
```

```
            are deleted and initialized*/
////////////////////////////////////////////////////////////////
///////////////
void CGraphWin::InitRelationViews()
{
  CView* item;
  CIterator doTo(itsScroller->GetSubviews());
  while (item = (CView*) doTo.Next())
    if (item->GetId()==RELATION_VIEW_ID)
      delete (CRelationLine *)item;
  CWindow::DoCommand(VIEWRelationCmd);
  UpdateRelationViews();
}
////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
    Usage: This is used to Initialize Relation Views
           of theGraphNodeId
    Work Done: Relation Views of theGRaphNodeId are
           initialized*/
////////////////////////////////////////////////////////////////
///////////
void CGraphWin::InitRelationViews(int theGraphNodeId)
{
  char charArray[20];
  CString aMsgString;
  sprintf(charArray, "%d", theGraphNodeId);
  aMsgString = charArray;
  CWindow::DoCommand(VIEWRelationCmd, &aMsgString);
}
////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
    Usage: This is used to Update Relation Views
    Work Done: Views are searched from itsScroller
           All Views of type RELATION_VIEW_ID
           are selected and made shown or hidden*/
////////////////////////////////////////////////////////////////
///////////
void CGraphWin::UpdateRelationViews()
{
CView* item;
CIterator doTo(itsScroller->GetSubviews());
while (item = (CView*) doTo.Next())
  if (item->GetId()==RELATION_VIEW_ID)
```

```
    {
    if (itsRelationsVisible)
      ((CRelationLine *)item)->Show();
    else
      ((CRelationLine *)item)->Hide();
    }
itsScroller->DoDraw();
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
    Usage: This is used to Update Relation Views
           of theGraphNodeId
    Work Done: Views are searched from itsScroller
           All Views of type RELATION_VIEW_ID
           are updated and made shown or hidden*/
////////////////////////////////////////////////////////////
///////////
void CGraphWin::UpdateRelationViews(int theGraphNodeId)
{
CView* item;
CIterator doTo(itsScroller->GetSubviews());
while (item = (CView*) doTo.Next())
  if (item->GetId()==RELATION_VIEW_ID)
    {
    if( (((CRelationLine *)item)-
>Update(theGraphNodeId))&&
                          (itsRelationsVisible) )
      ((CRelationLine *)item)->Show();
    else
      ((CRelationLine *)item)->Hide();
    }
if (itsRelationsVisible) itsScroller->DoDraw();
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
    Usage: This is used to Delete Relation Views
    Work Done: Views are searched from itsScroller
           All Views of type RELATION_VIEW_ID
           are deleted */
////////////////////////////////////////////////////////////
///////////
void CGraphWin::DeleteRelationViews(void)
{
```

```
CView* item;
CIterator doTo(itsScroller->GetSubviews());
while (item = (CView*) doTo.Next())
   if (item->GetId()==RELATION_VIEW_ID)
      delete (CRelationLine *)item;
if (itsRelationsVisible) itsScroller->DoDraw();
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
     Usage: This is used to Delete Relation Views
            of theGraphNodeId
     Work Done: Relation Views of theGraphNodeId are
            searched from itsScroller.
            All Views of type RELATION_VIEW_ID
            are deleted*/
////////////////////////////////////////////////////////////
///////////////
void CGraphWin::DeleteRelationViews(int theGraphNodeId)
{
CView* item;
CIterator doTo(itsScroller->GetSubviews());
while (item = (CView*) doTo.Next())
   if (item->GetId()==RELATION_VIEW_ID)
      {
      if(((CRelationLine *)item)->Update(theGraphNodeId))
        delete (CRelationLine *)item;
      }
if (itsRelationsVisible) itsScroller->DoDraw();
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
     Usage: This is used to make NewRelation
     Work Done: Information about relations is structured
            and initialized  */
////////////////////////////////////////////////////////////
///////////
void CGraphWin::NewRelation(int theFirstRelationId, int
theOtherRelationId,
                int theFirstRelPos, int theOtherRelPos,
                CString theFirstRelString,
                CString theOtherRelString)
{
    char aMsg[200];
```

```
  CString aString;
  sprintf(aMsg, "%d ,%d ,%d
,%s",theFirstRelationId,theFirstRelPos,

  theOtherRelationId,theFirstRelString.GetCharPtr());
  aString = aMsg;
  DoCommand(UPDATERelationInfoCmd, &aString);
  RelationName(theFirstRelationId, theFirstRelString);
  if (theFirstRelationId != theOtherRelationId)
    {
    sprintf(aMsg, "%d ,%d ,%d
,%s",theOtherRelationId,theOtherRelPos,

  theFirstRelationId,theOtherRelString.GetCharPtr());
    aString = aMsg;
    DoCommand(UPDATERelationInfoCmd, &aString);
    RelationName(theOtherRelationId, theOtherRelString);
    }
}

//////////////////////////////////////////////////////////////
//////////
/******          Description              **********
    Usage: This is used to register a Relation Name
    Work Done: Information about relation name is
           initialized in itsRelationNameList
           and set in as menu item with
           theRelationId*/
//////////////////////////////////////////////////////////////
//////////
void CGraphWin::RelationName(int theRelationId, CString
theRelationString)
{
  itsRelationNameList.Replace(theRelationString,
theRelationId);
  win_set_menu_text(GetXVTWindow(), theRelationId,
                    theRelationString.GetCharPtr());
}
//////////////////////////////////////////////////////////////
//////////
/******          Description              **********
    Usage: This is used to make Relation Draw itself
    Work Done: CRelationLine used to Draw Relation
           is initialized*/
```

```
//////////////////////////////////////////////////////////////////
//////////
void CGraphWin::RelationDraw(int theFirstNodeId, int
theOtherNodeId)
{
CRelationLine *aRelationLine;
CRect aRect;
aRelationLine = new CRelationLine(itsScroller, aRect,
                        theFirstNodeId, theOtherNodeId);
aRelationLine->IRelationLine(itsRelationsVisible);
}
```

```
/*****************************************************************
*********
 *
```

CGraphNode.h

```
   File Name:    CGraphNode.h
                 CGraphNd.h  (DOS name)
   Author:       Sanjeev K. Guleria
   Date:      Wed Apr 13 11:13:52:32 IST 1994

   Class:    CGraphNode
   Inheritance:     CBoss
   Helper(s):

   Purpose: This is the class derived from CBoss. It
   serves the purpose of GraphNode in Graph.

   Usage:You may rename this class and add or override
       methods as nessesary.

   Override:

   Modifications:
                       *

*****************************************************************
********/
#ifndef CGraphNode_H
#define CGraphNode_H

#include "PwrDef.h"
#include CBoss_i
class CGraphDoc;
class CGraphNode : public CBoss
{
  public:
   CGraphNode(CGraphDoc * theGraphDoc, int theId, int
theViewer);
   //constructor
   virtual void DoCommand(long theCommand,void*
theData=NULL);

   //void SetProcess(int theNewGraphProcessType, void *
theNewGraphProcess);
   /*Sets the process associated with the node*/
```

```
  CString GetTitle();
  void NewRelation(CString theString, int theRelationType,
BOOLEAN theNormalMsg);
  void SendRelation(long theCommand, int theRelationType,
BOOLEAN theNormalMsg);
  BOOLEAN UnSetRelation(int theOtherGraphNodeId, int
theRelationType);
  /*returns TRUE if parent-child relation existed. UnSets
the
  parent-child relation between these two CGraphNode
objects*/
  void ExecuteProcess(void);
 protected:
  int GetId();
  //CString GetTitle();
  CGraphDoc * itsGraph;
  int GetRelation(int theOtherGraphNodeId, CString
theRelationString);
  int GetRelation(int theOtherGraphNodeId, int
theRelationType);
  /*returns number of times this node is related to
theOtherGraphNode */
  int GetRelationPosition(int theOtherGraphNodeId, CString
theRelationString);
  int GetRelationPosition(int theOtherGraphNodeId, int
theRelationType);
  /*returns the position at which entry of relation exists
as relation to theOtherGraphNode */
  CString SetRelation(int theOtherGraphNodeId, CString
theRelationString);
  void SetRelation(int theOtherGraphNodeId, int
theRelationType);
  /*returns TRUE if theOtherId is different and that no
relation of this type
  exists with it. Sets the specified relation*/
  BOOLEAN UnSetRelation(int theOtherGraphNodeId, CString&
theRelationString);
  CString GetRelationString(int theGraphNodeId, int
theRelationType);
  /*Gets a list of Id's related to theGraphNodeId by
theParentOrChild Relation*/
  void SetRelationString(int theGraphNodeId, int
theRelationType,
                         CString theString);
```

```
  ////////////////////////////////////////////////////////////
////////////////////
  void ClearAllRelations(int theGraphNodeId, int
theRelationType);
  /*removes all graph nodes theRelationType tree*/
  private:
  int itsGraphNodeId;     //Id of the graph node
  int itsViewer;
  void GiveMessage(long theCommand, int theOtherNodeId,
          int theRelationType, BOOLEAN theNormalMsg=TRUE);
};

#endif //CGraphNode_H
```

```
/*****************************************************************
**********
 *
```

CGraphNode.cxx

```
   File Name:    CGraphNode.cxx
                 CGraphNd.cpp   (DOS name)
   Author:       Sanjeev K. Guleria
   Date:      Wed Apr 13 11:13:52:32 IST 1994

   Modifications:
                          *

*****************************************************************
*********/
#include "windows.h"
#ifdef DOS
#include "CGraphDc.h"
#include "CGraphNd.h"
#else
#include "CGrapherDoc.h"
#include "CGrapherNode.h"
#endif
#include "PwrDef.h"
#include CWindow_i
#include CString_i
#include CView_i
#include "CUtility.h"
#include "Grapher.h"

//////////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
Usage: This constructs the instance of CGraphNode.
Work done:
* The attributes are initialized.                    */
//////////////////////////////////////////////////////////////////
//////////////////
CGraphNode::CGraphNode(CGraphDoc * theGraphDoc, int theId,
int theViewer)
{
itsGraphNodeId = theId;
itsGraph = theGraphDoc;
itsViewer = theViewer;
if (itsViewer)
```

```
  xvt_note("The {%s<<[%d]>>%s}Node (%s) was clicked in Graph
(%s)",
  GetRelationString(itsGraphNodeId,
PARENT_RELATION).GetCharPtr(),
  itsGraphNodeId,
  GetRelationString(itsGraphNodeId,
CHILD_RELATION).GetCharPtr(),
  GetTitle().GetCharPtr(),
  (itsGraph->FindWindow(itsGraph->GetId())-
>GetTitle()).GetCharPtr());
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
    Usage: This is to Get Id  */
////////////////////////////////////////////////////////////////
//////////////
int CGraphNode::GetId()
{
return itsGraphNodeId;
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
    Usage: This is to GetTitle  */
////////////////////////////////////////////////////////////////
//////////////
CString CGraphNode::GetTitle()
{
char charMsg[20];
CString aString;
sprintf(charMsg,"%d ",GetId());
aString = charMsg;
DoCommand(GETNodeTitleCmd, &aString);
return (aString);
}
////////////////////////////////////////////////////////////////
//////////////////
/******          Description              **********
    Usage: This is CGraphDoc level DoCommand handler  */
////////////////////////////////////////////////////////////////
//////////////
void CGraphNode::DoCommand(long theCommand,void* theData)
{
itsGraph->DoCommand(theCommand, theData);
```

```
}
//////////////////////////////////////////////////////////////
//////////////////
/******          Description            **********
    Usage: This is used to get the number of times
theOtherGraphNodeId
           is defined in theRelationString
    Work Done: For this the appropriate list is extracted
           and the count is get from there.
           That count is returned*/
//////////////////////////////////////////////////////////////
///////////////
int CGraphNode::GetRelation(int theOtherGraphNodeId, CString
theRelationString)
{
int intTotalRelations =
atoi(theRelationString.GetCharPtr());
char *charPtrTemp;
int intTemp, intNumChars, intNumRelations = 0;
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
  {
  charPtrTemp =
aUtility.GetNthList(theRelationString.GetCharPtr(),
ITEM_SEPARATOR,
                              aCount, &intNumChars);
  intTemp = atoi(charPtrTemp);
  if (intTemp == theOtherGraphNodeId)
    intNumRelations++;
  }
return (intNumRelations);
}
//////////////////////////////////////////////////////////////
////////////////
int CGraphNode::GetRelation(int theOtherGraphNodeId, int
theRelationType)
{
CString aString(GetRelationString(GetId(),
theRelationType));
return (GetRelation(theOtherGraphNodeId, aString));
}
//////////////////////////////////////////////////////////////
//////////////////
```

```
/******          Description              **********
     Usage: This is used to get the relation position
            in theRelationString
     Work Done: For this the appropriate list is extracted
            and the relation position is get from there.
            That position is returned*/
////////////////////////////////////////////////////////////
///////////////
int CGraphNode::GetRelationPosition(int theOtherGraphNodeId,
CString theRelationString)
{
int intTotalRelations =
atoi(theRelationString.GetCharPtr());
char *charPtrTemp;
int intTemp, intNumChars, intRelationPosition = 0;
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
   {
   charPtrTemp =
aUtility.GetNthList(theRelationString.GetCharPtr(),
ITEM_SEPARATOR,
                              aCount, &intNumChars);
   intTemp = atoi(charPtrTemp);
   if (intTemp == theOtherGraphNodeId)
      {
      intRelationPosition = aCount;
      break;
      }
   }
return (intRelationPosition);
}
////////////////////////////////////////////////////////////
///////////////////
/******          Description              **********
     Usage: This is used to get the relation position
            of  theRelationType
     Work Done: For this the appropriate list is extracted
            and the relation position is returned*/
////////////////////////////////////////////////////////////
///////////////
int CGraphNode::GetRelationPosition(int theOtherGraphNodeId,
int theRelationType)
{
```

```
CString aString(GetRelationString(GetId(),
theRelationType));
return (GetRelationPosition(theOtherGraphNodeId, aString));
}
////////////////////////////////////////////////////////////
////////////////
/******          Description            **********
     Usage: This is used to Set Relation in
theRelationString
     Work Done: theOtherGraphNodeId is added there and the
count
           is incremented by one*/
////////////////////////////////////////////////////////////
////////////////
CString CGraphNode::SetRelation(int theOtherGraphNodeId,
CString theRelationString)
{
int intTotalRelations =
atoi(theRelationString.GetCharPtr());
char charArray[20];
char * charPtrTemp;
CUtility aUtility;
sprintf(charArray, "%d %d", intTotalRelations+1,
theOtherGraphNodeId);

int intLength = theRelationString.Length() +
strlen(charArray) + 1;
charPtrTemp = new char[intLength];

aUtility.ReplaceNthGivenList(theRelationString.GetCharPtr(),
ITEM_SEPARATOR,
                    1, charArray, charPtrTemp);
theRelationString = charPtrTemp;
delete[] charPtrTemp;
return theRelationString;
}
////////////////////////////////////////////////////////////
////////////////
/******          Description            **********
     Usage: This is used to Set Relation with
theOtherGraphNodeId
     Work Done: Firstly the relation list is extracted and
           relation is set in that string. After that it
           the string is updated */
```

```
////////////////////////////////////////////////////////////////
////////////////
void CGraphNode::SetRelation(int theOtherGraphNodeId, int
theRelationType)
{
CString aString(GetRelationString(GetId(),
theRelationType));
aString = SetRelation(theOtherGraphNodeId, aString);
SetRelationString(GetId(), theRelationType, aString);
}
////////////////////////////////////////////////////////////////
////////////////
/******          Description            **********
    Usage: This is used to Un Set Relation in
theRelationString
    Work Done: Firstly theOtherGraphNodeId is searched in
           theRelationString. If its position is found
           then it is removed from there and the count
           is decremented by one*/
////////////////////////////////////////////////////////////////
////////////////
BOOLEAN CGraphNode::UnSetRelation(int theOtherGraphNodeId,
CString& theRelationString)
{
int intTotalRelations =
atoi(theRelationString.GetCharPtr());
int intRelationPosition = 0;
char charArray[20];
char * charPtrTemp;
CUtility aUtility;
BOOLEAN aResult = FALSE;
if (intRelationPosition =
GetRelationPosition(theOtherGraphNodeId, theRelationString))
  {
  int intLength = theRelationString.Length() + 1;
  charPtrTemp = new char[intLength];
  aUtility.ReplaceNthGivenList(theRelationString.GetCharPtr
(), ITEM_SEPARATOR,
              intRelationPosition, "", charPtrTemp);
  sprintf(charArray, "%d", intTotalRelations-1);
  aUtility.ReplaceNthGivenList(charPtrTemp, ITEM_SEPARATOR,
              1, charArray, charPtrTemp);
    theRelationString = charPtrTemp;
  delete[] charPtrTemp;
  aResult = TRUE;
```

```
   }
return(aResult);
}
////////////////////////////////////////////////////////////////
////////////////
/******          Description            **********
     Usage: This is used to form Un Set Relation
     Work Done: Firstly the relation list is extracted and
            relation is un set in that string. If it
            is unset there then the string is updated*/
////////////////////////////////////////////////////////////////
////////////////
BOOLEAN CGraphNode::UnSetRelation(int theOtherGraphNodeId,
                        int theRelationType)
{
CString aString(GetRelationString(GetId(),
theRelationType));
BOOLEAN aResult = FALSE;
if (aResult = UnSetRelation(theOtherGraphNodeId, aString))
   {
   SetRelationString(GetId(), theRelationType, aString);
   }
return(aResult);
}
////////////////////////////////////////////////////////////////
////////////
/******          Description            **********
     Usage: This is used to form New Relation
     Work Done: Firstly it is checked that any of
            the relations do not exist previously
            so that the relation can be formed
            the information is passed to send
            message in packets*/
////////////////////////////////////////////////////////////////
////////////////
void CGraphNode::NewRelation(CString theString, int
theRelationType,
            BOOLEAN theNormalMsg)
{
CString aString(GetRelationString(GetId(),
theRelationType));
int intTotalRelations = atoi(theString.GetCharPtr());
int intTemp, intNumChars, intRelationPosition = 0;
char * charPtrTemp;
BOOLEAN aChange = FALSE;
```

```
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
   {
   charPtrTemp = aUtility.GetNthList(theString.GetCharPtr(),
ITEM_SEPARATOR,
                                aCount, &intNumChars);
   intTemp = atoi(charPtrTemp);
     if ( (GetId() != intTemp) && !GetRelation(intTemp,
aString))
      {
      aString = SetRelation(intTemp, aString);
      GiveMessage(SENDRelationCmd, intTemp, theRelationType,
theNormalMsg);
      aChange = TRUE;
      }
     }
if (aChange) SetRelationString(GetId(), theRelationType,
aString);
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
     Usage: This is used to Send Relation Information
     Work Done: the information is passed to send message
            in packets*/
////////////////////////////////////////////////////////////
////////////////
void CGraphNode::SendRelation(long theCommand, int
theRelationType,
         BOOLEAN theNormalMsg)
{
CString aString(GetRelationString(GetId(),
theRelationType));
int intTotalRelations = atoi(aString.GetCharPtr());
char *charPtrTemp;
int intTemp, intNumChars, intNumRelations = 0;
CUtility aUtility;

for(int aCount = 2; (aCount < (intTotalRelations + 2) );
aCount++)
   {
   charPtrTemp = aUtility.GetNthList(aString.GetCharPtr(),
ITEM_SEPARATOR,
```

```
                                aCount, &intNumChars);
   intTemp = atoi(charPtrTemp);
   GiveMessage(theCommand, intTemp, theRelationType,
theNormalMsg);
   }
}
////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to Get Relation String
     Work Done: Parameter information is passed to
           supervisor  and relation is returned*/
////////////////////////////////////////////////////////////////
///////////////
CString CGraphNode::GetRelationString(int theGraphNodeId,
int theParentOrChild)
{
return itsGraph->GetInfoList(theGraphNodeId,
theParentOrChild);
}
////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to Set Relation String
     Work Done: Parameter information is passed to
supervisor  */
////////////////////////////////////////////////////////////////
///////////////
void CGraphNode::SetRelationString(int theGraphNodeId, int
theParentOrChild, CString theString)
{
itsGraph->SetInfoList(theGraphNodeId, theParentOrChild,
theString);
}
////////////////////////////////////////////////////////////////
///////////
/******          Description              **********
     Usage: This is used to Give Message to supervisor
     Work Done: Parameter information is structured and
           sent to supervisor */
////////////////////////////////////////////////////////////////
///////////////
void CGraphNode::GiveMessage(long theCommand, int
theOtherNodeId,
             int theRelationType, BOOLEAN theNormalMsg)
```

```
{
char charMsg[20];
CString aMsgString;
  if (theNormalMsg)
    sprintf(charMsg,"%d ,%d, %d",GetId(),theOtherNodeId,
theRelationType);
  else
    sprintf(charMsg,"%d ,%d, %d",theOtherNodeId,GetId(),
theRelationType);
  aMsgString = charMsg;
  DoCommand(theCommand, &aMsgString);
}
////////////////////////////////////////////////////////////
///////////
/******          Description            **********
    Usage: This is used to Execute Process
    Work Done: Process information is get from supervisor
           and executed accordingly  */
////////////////////////////////////////////////////////////
///////////
void CGraphNode::ExecuteProcess(void)
{
CString aProcessString = GetRelationString(GetId(),
PROCESS);
char *charPtrTemp;
CUtility aUtility;
charPtrTemp =
aUtility.GetNthList(aProcessString.GetCharPtr(),
                              ITEM_SEPARATOR, 2, NULL);
switch (atoi(aProcessString.GetCharPtr()))
  {
  case PROCESS_FUNCTION:
      DoCommand(atoi(charPtrTemp));
      break;
  case PROCESS_EXEC:
    WinExec(charPtrTemp, SW_SHOW);
break;
  }
}
```

```
/*****************************************************************
*********
 *
```

CAskDlg.h

```
   File Name:    CAskDlg.h
   Author:     Sanjeev K. Guleria
   Date:     Tue Apr 5 17:38:58:32 IST 1994

   Class:   CAskDlg
   Inheritance:          CModalDialog
   Helper(s):

   Purpose:         This Dialog is used to get some data in
the form of
              a string form the user through the Interface

                        *

*****************************************************************
********/

#ifndef CAskDlg_H
#define CAskDlg_H

#include "PwrNames.h"
#include "PwrDef.h"
#include CDialog_i

class CAskDlg : public CModalDialog
{
 public:

  friend CString AskDlg();
  friend CString AskDlg(CString * theResponse);
 private:

  CString* itsResponse;

  CAskDlg(CString* theResponse);
  void DoCreate(WINDOW theXVTWindow);
  virtual void DoButton(int theControlId);
};

#endif //CAskDlg_H
```

```
/*****************************************************************
**********
 *
```

CAskDlg.cxx

```
   File Name:    CAskDlg.cxx
   Author:     Sanjeev K. Guleria
   Date:     Tue Apr 4 18:08:20:54 IST 1994

   Modifications:
                          *

*****************************************************************
*********/

#include "CAskDlg.h"
#include "Grapher.h"
#include CString_i

//////////////////////////////////////////////////////////////////
///////////
CString AskDlg()
{
  // This function places the AskDlg dialog and returns the
string entered.

  CString theResponse;
  CModalDialog* dlg = new CAskDlg(&theResponse);
  dlg->DoModal();
  delete dlg;
  return theResponse;
}
//////////////////////////////////////////////////////////////////
///////////
CString AskDlg(CString* theResponse)
{
  // This function places the AskDlg dialog and returns the
string entered.

  CModalDialog* dlg = new CAskDlg(theResponse);
```

```
  dlg->DoModal();
  delete dlg;
  return *theResponse;
}
////////////////////////////////////////////////////////////
///////////
CAskDlg::CAskDlg(CString* theResponse)

  : CModalDialog(NULL, GETTEXTBOXRid),
itsResponse(theResponse)
{
}
void CAskDlg::DoCreate(WINDOW theXVTWindow)
{
  CDialog::DoCreate(theXVTWindow);
  WINDOW theEdit = FindXVTControl(GETTextEdit);
  set_title(theEdit, itsResponse->GetCharPtr());
}
void CAskDlg::DoButton(int theControlId)
{
  // OK button has been pressed: Get edit box text and close
dialog.

  WINDOW theEdit = FindXVTControl(GETTextEdit);

  switch(theControlId)
  {
  case DLG_OK:
  {
  char text[100];
  *itsResponse = get_title(theEdit, text, 100);
  if (itsResponse->Length()>0)
  {
  Close();
  }
  else xvt_note("Can\'t Let you go like this");
  break;
  }
  case DLG_CANCEL:
  Close();
  break;
  }
}
```

```
/*****************************************************************
*********
 *
```

CRelDlg.h

```
   File Name:    CRelDlg.h
   Author:      Sanjeev K. Guleria
   Date:    Wed Apr 6 12:42:38 IST 1994

   Class:   CRelDlg
   Inheritance:          CModalDialog
   Helper(s):

   Purpose:

                        *

*****************************************************************
********/

#ifndef CRelDlg_H
#define CRelDlg_H

#include "PwrNames.h"
#include "PwrDef.h"
#include CDialog_i

class CRelDlg : public CModalDialog
{
 public:

  friend void RelDlg(CString * theFirstResponse, CString *
theOtherResponse);
 private:

  CString* itsFirstResponse;
  CString* itsOtherResponse;

  CRelDlg(CString * theFirstResponse, CString *
theOtherResponse);
  void DoCreate(WINDOW theXVTWindow);
  virtual void DoButton(int theControlId);
};

#endif //CRelDlg_H
```

```
/*****************************************************************
**********
 *
```

CRelDlg.cxx

```
   File Name:    CRelDlg.cxx

   Modifications:
                          *

*****************************************************************
*********/

#include "CRelDlg.h"
#include "Grapher.h"
#include CString_i

//////////////////////////////////////////////////////////////////
//////////
void RelDlg(CString * theFirstResponse, CString *
theOtherResponse)
{
  // This function places the RelDlg dialog and returns the
string entered.

  CModalDialog* dlg = new CRelDlg(theFirstResponse,
theOtherResponse);
  dlg->DoModal();
  delete dlg;
}
//////////////////////////////////////////////////////////////////
//////////
CRelDlg::CRelDlg(CString * theFirstResponse, CString *
theOtherResponse)

  : CModalDialog(NULL, GETRELATIONBOXRid),
  itsFirstResponse(theFirstResponse),
  itsOtherResponse(theOtherResponse)
{
}
```

```
void CRelDlg::DoCreate(WINDOW theXVTWindow)
{
  CDialog::DoCreate(theXVTWindow);
  WINDOW theEdit = FindXVTControl(GETRelationDlgEdit1);
  set_title(theEdit, itsFirstResponse->GetCharPtr());
  theEdit = FindXVTControl(GETRelationDlgEdit2);
  set_title(theEdit, itsOtherResponse->GetCharPtr());
}
void CRelDlg::DoButton(int theControlId)
{
  // OK button has been pressed: Get edit box text and close
dialog.

  WINDOW theEdit1 = FindXVTControl(GETRelationDlgEdit1);
  WINDOW theEdit2 = FindXVTControl(GETRelationDlgEdit2);

  switch(theControlId)
  {
    case DLG_OK:
      {
      char text[100];
      int intIndex = 0;
      *itsFirstResponse = get_title(theEdit1, text, 100);
      intIndex = strspn(text, " \t");
      //if ((strlen(text) - intIndex) >0)
       *itsFirstResponse = &text[intIndex];
      *itsOtherResponse = get_title(theEdit2, text, 100);
      intIndex = strspn(text, " \t");
      //if ((intIndex = strlen(text) - strspn(text, " \t"))
>0)
       *itsOtherResponse = &text[intIndex];
      if ((itsFirstResponse->Length()>0) &&
(itsOtherResponse->Length()>0))
       {
       Close();
       }
      else
       xvt_note("Can\'t Let you go like this");
      break;
      }
  }
}
```

```
/*****************************************************************
*********
 *
```

CGraphRectangle.h

```
   File Name:           CGraphRectangle.h
                CGrphRct.h       (DOS)
   Author:     Sanjeev K. Guleria
   Date:     Wed Apr 11 12:55:00:02 IST 1994

   Class:               CGraphRectangle
   Inheritance:         CRectangle
   Helper(s):           CText

   Purpose:       This takes care of the view level of
GraphNode
            represented as a rectangle with name inside.

   Usage:         You may rename this class and add overriden
methods as
      nessesary.

   Override:

   Modifications:

*****************************************************************
********/
#ifndef CGraphRectangle_H
#define CGraphRectangle_H
#include "Grapher.h"
#include "PwrDef.h"
#include CRectangle_i
#include CText_i
class CGraphRectangle : public CRectangle
{
  public:
     CGraphRectangle(CSubview* theEnclosure, const CRect&
theRegion,
        const CString& theTitle);    //constructor
     virtual CView* FindEventTarget(const CPoint&
theLocation) const;
   virtual void Draw(const CRect& theClippingRegion);
  private:
CText itsText;
```

```
};

#endif //CGraphRectangle_H
```

```
/*****************************************************************
*********
 *
```

CGraphRectangle.cxx

```
   File Name:           CGraphRectangle.cxx
                CGrphRct.cpp     (DOS)
   Author:     Sanjeev K. Guleria
   Date:    Wed Apr 11 12:55:00:02 IST 1994


*****************************************************************
********/
#include "CGrphRct.h"

CGraphRectangle::CGraphRectangle(CSubview* theEnclosure,
const CRect& theRegion,
       const CString& theTitle):CRectangle(theEnclosure,
theRegion)
       ,itsText(theEnclosure, CPoint(0,0),theTitle)
{
SetTitle(theTitle);
itsText.SetEnclosure(this);
}
//////////////////////////////////////////////////////////////////
//
void CGraphRectangle::Draw(const CRect& theClippingRegion)
{
  CRectangle::Draw(theClippingRegion);
  itsText.SetText(GetTitle());
  CRect aGraphRect = GetFrame();
  CRect aTextRect;
  aTextRect.Top((aGraphRect.Height()-
itsText.GetHeight())/2);
  aTextRect.Left((aGraphRect.Width()-
itsText.GetWidth())/2);
  aTextRect.Height(itsText.GetHeight());
  aTextRect.Width(itsText.GetWidth());
  itsText.Size(aTextRect);
  itsText.Draw(theClippingRegion);

}
//////////////////////////////////////////////////////////////////
//
```

```
CView* CGraphRectangle::FindEventTarget(const CPoint&
theLocation) const
{
return (CView*)this;
}
```

CRelationLine.h

```
/*****************************************************************
*********
 *
   File Name:    CRelationLine.h
              CRelLine.h     (DOS)
   Author:      Sanjeev K. Guleria
   Date:    Sat Apr 9 13:40:53 IST 1994

   Class:   CRelationLine
   Inheritance:          CShape
   Helper(s):

   Purpose:

                        *

*****************************************************************
********/
#ifndef CRelationLine_H
#define CRelationLine_H
#include "Grapher.h"
#include "PwrDef.h"
#include CLine_i
#include CShape_i
#include CWindow_i
class CRelationLine : public CShape
{
  public:
     CRelationLine(CSubview* theEnclosure, const CRect&
theRegion,
     int theFirstId, int theOtherId);    //constructor
     ~CRelationLine();
     BOOLEAN IRelationLine(BOOLEAN isVisible = TRUE,
GLUETYPE itsGlue = NULLSTICKY);
     BOOLEAN Update(int theId);
     virtual void Draw(const CRect& theClippingRegion);
     virtual void Show(void);
     virtual void Hide(void);
     int GetFirstId();
     int GetOtherId();
  private:
int itsFirstViewId ;
int itsOtherViewId ;
```

```
CLine itsHalfArrowLine;
CLine itsOtherHalfLine;
};

#endif //CRelationLine_H

/*****************************************************************
**********
 *
```

CRelationLine.cxx

```
   File Name:    CRelationLine.cxx
              CRelLine.cpp     (DOS)
   Author:       Sanjeev K. Guleria
   Date:     Sat Apr 9 16:07:20 IST 1994


   Modifications:
                          *

*****************************************************************
*********/
#include "CRelLine.h"
#include "grapher.h"
//////////////////////////////////////////////////////////////////
//
CRelationLine::CRelationLine(CSubview* theEnclosure, const
CRect& theRegion,
     int theFirstId, int theOtherId):CShape(theEnclosure,
theRegion)
     ,itsHalfArrowLine(theEnclosure, CPoint(0,0),
CPoint(0,0))
     ,itsOtherHalfLine(theEnclosure, CPoint(0,0),
CPoint(0,0))
{
itsFirstViewId = theFirstId;
itsOtherViewId = theOtherId;
}
//////////////////////////////////////////////////////////////////
//
CRelationLine::~CRelationLine()
{
}
//////////////////////////////////////////////////////////////////
//
```

```
int CRelationLine::GetFirstId()
{
return itsFirstViewId;
}
////////////////////////////////////////////////////////////
//
int CRelationLine::GetOtherId()
{
return itsOtherViewId;
}
////////////////////////////////////////////////////////////
//
BOOLEAN CRelationLine::IRelationLine(BOOLEAN isVisible,
GLUETYPE itsGlue)
{
itsHalfArrowLine.SetEnclosure(this);
itsOtherHalfLine.SetEnclosure(this);
itsHalfArrowLine.ILine(FALSE,TRUE,isVisible);
itsOtherHalfLine.ILine(FALSE,FALSE,isVisible);
Update(itsFirstViewId);
SetId(RELATION_VIEW_ID);
return CShape::IShape(isVisible, itsGlue);
}
////////////////////////////////////////////////////////////
//
void CRelationLine::Show()
{
CShape::Show();
itsHalfArrowLine.Show();
itsOtherHalfLine.Show();
}
////////////////////////////////////////////////////////////
//
void CRelationLine::Hide()
{
CShape::Hide();
itsHalfArrowLine.Hide();
itsOtherHalfLine.Hide();
}
////////////////////////////////////////////////////////////
//
void CRelationLine::Draw(const CRect& theClippingRegion)
{
itsHalfArrowLine.Draw(theClippingRegion);
itsOtherHalfLine.Draw(theClippingRegion);
```

```
}
////////////////////////////////////////////////////////////////
//
BOOLEAN CRelationLine::Update(int theId)
{
if ((theId==itsFirstViewId)||(theId==itsOtherViewId))
   {
   CRect aFirstRect, anOtherRect;
   CPoint aFirstPoint, aMidPoint, anOtherPoint;

   aFirstRect = GetEnclosure()->FindSubview(itsFirstViewId)-
>GetFrame();
   aFirstPoint.SetPoint((aFirstRect.Left()+aFirstRect.Right(
))/2,(aFirstRect.Top()+aFirstRect.Bottom())/2);
   anOtherRect = GetEnclosure()-
>FindSubview(itsOtherViewId)->GetFrame();
   anOtherPoint.SetPoint((anOtherRect.Left()+anOtherRect.Rig
ht())/2,(anOtherRect.Top()+anOtherRect.Bottom())/2);
   aMidPoint.SetPoint((aFirstPoint.H()+anOtherPoint.H())/2,(
aFirstPoint.V()+anOtherPoint.V())/2);

   CRect aRect(aFirstPoint, anOtherPoint);
   itsHalfArrowLine.Size(aFirstPoint, aMidPoint);
   itsOtherHalfLine.Size(aMidPoint, anOtherPoint);
   Size(aRect);

   return TRUE;
   }
else return FALSE;
}
```

```
/*****************************************************************
****************
```

CUtility.h

```
File    CUtility.h

class     CUtility

Author    Sanjeev K. Guleria

Dated     21st March, 1994.

Contains  Utilities to handle null terminated Strings

*****************************************************************
***************/
#ifndef CUtility_H
#define CUtility_H

#include "xvt.h"
#include "string.h"
#include "stdlib.h"
class CUtility
{
public:
CUtility(void);
char *CutGivenList(char * str, char *theSeparatorList,  char
*theGivenString);
char *GetNthList(char * str, char *theSeparatorList,  int
thePosition, int * theNumChars);
char *GetAfterNthList(char * str, char *theSeparatorList,
int thePosition);
char *NullTerminateNthList(char * str, char
*theSeparatorList,  int thePosition);
char *GetNthNullTerminatedList(char * str, int thePosition);
char *ReplaceNthGivenList(char * str, char
*theSeparatorList,
            int thePosition, char *theGivenString, char
*theBuffer);
char *NullTerminateUptoNthList(char * theOutputString, char
*theSeparatorList,  int thePosition);
private:
char *GivenList(char * theOutputString, char
*theSeparatorList,  char *theGivenString );
};
```

```
#endif //CUtility_H
```

```
/*****************************************************************
****************
```

CUtility.cpp

```
File    CUtility.cpp

class     CUtility

Author    Sanjeev K. Guleria

Dated     21st March, 1994.

Contains  Utilities to handle null terminated Strings

*****************************************************************
***************/

#include "CUtility.h"

////////////////////////////////////////////////////////////////
////////////////
//          Internal Function.
////////////////////////////////////////////////////////////////
////////////////
CUtility::CUtility(void)
{
}
char* CUtility::NullTerminateUptoNthList(char *
theOutputString, char *theSeparatorList,  int thePosition)
{
char *aTransitionString;
int length;
int aCount;
length = strlen(theOutputString);
if ( (thePosition <= 0) ||
    ( ((length+1)/2) < thePosition) )
  return NULL;
aTransitionString = theOutputString;
for(aCount = 1,
  aTransitionString = strtok(aTransitionString,
theSeparatorList);
  ( (aCount < thePosition) && (aTransitionString) );
  aCount++,
  aTransitionString = strtok(NULL, theSeparatorList))
    ;
```

```
return(aTransitionString);
}
////////////////////////////////////////////////////////////////
////////////////
//          Internal Function.
////////////////////////////////////////////////////////////////
////////////////
char * CUtility :: GivenList(char * theOutputString, char
*theSeparatorList,  char *theGivenString )
{
char *aTransitionString;
int length;
length = strlen(theOutputString);
if (!theGivenString)
  return NULL;
aTransitionString = theOutputString;
for(
  aTransitionString = strtok(aTransitionString,
theSeparatorList);
  ( (aTransitionString) && strcmp(theGivenString,
aTransitionString));
  aTransitionString = strtok(NULL, theSeparatorList))
    ;
return(aTransitionString);
}
////////////////////////////////////////////////////////////////
////////////////
//     Use    :    Use this function to cut theGivenString
from str.
////////////////////////////////////////////////////////////////
////////////////
char * CUtility :: CutGivenList(char * str, char
*theSeparatorList,  char *theGivenString)
{
char *aTransitionString;
char *anOutputString;
int length;
int difference;
length = strlen(str)+1;
anOutputString = (char *)xvt_malloc(length);
strcpy(anOutputString, str);

aTransitionString = GivenList(anOutputString,
theSeparatorList, theGivenString);
difference = aTransitionString-anOutputString;
```

```
length = strlen(aTransitionString);
if (aTransitionString)
   {
   aTransitionString = NULL;
   if (difference)
     aTransitionString = &str[length];
   else
     {
     if (strlen(str) == (unsigned)(difference+length))
         {
       str[difference] = '\0' ;
         }
       else
         {
         strcpy( &str[difference], &str[difference+length])
;
         }
       aTransitionString = str;
       }
    }
xvt_free (anOutputString);
return(aTransitionString);
}
////////////////////////////////////////////////////////////
////////////////
//
//   Use  : Use this function to get Nth List separated by
any of
//          characters from theSeparatorList. After this
operation
//          str contains the whole list upto Nth List.
//
////////////////////////////////////////////////////////////
////////////////
char * CUtility :: GetNthList(char * str, char
*theSeparatorList,  int thePosition, int * theNumChars)
{
char *aTransitionString;
char *anOutputString;
int length;
int difference;
length = strlen(str)+1;
anOutputString = /*(char *)*/xvt_malloc(length);
strcpy(anOutputString, str);
```

```
aTransitionString = NullTerminateUptoNthList(anOutputString,
theSeparatorList, thePosition);
difference = aTransitionString-anOutputString ;
if (aTransitionString)
   {
   length = strlen(aTransitionString);
   aTransitionString = NULL;
   if (theNumChars)
     *theNumChars = length;
   //str[(difference)+strlen(aTransitionString)] = '\0' ;
   aTransitionString = &str[difference] ;
    }
xvt_free(anOutputString);
return(aTransitionString);
}
////////////////////////////////////////////////////////////
////////////////
//
//   Use  : Use this function to get Nth List separated by
any of
//          characters from theSeparatorList. After this
operation
//          str contains the whole list upto Nth List.
//
////////////////////////////////////////////////////////////
////////////////
char * CUtility :: NullTerminateNthList(char * str, char
*theSeparatorList,  int thePosition)
{
char *aTransitionString;
char *anOutputString;
int length;
int difference;
length = strlen(str)+1;
anOutputString = (char *)xvt_malloc(length);
strcpy(anOutputString, str);

aTransitionString = NullTerminateUptoNthList(anOutputString,
theSeparatorList, thePosition);
difference = aTransitionString-anOutputString ;

if (aTransitionString)
   {
   length = strlen(aTransitionString);
   str[(difference)+length] = '\0' ;
```

```
   aTransitionString = NULL;
   aTransitionString = &str[(difference)] ;
     }
xvt_free (anOutputString);
return(aTransitionString);
}
////////////////////////////////////////////////////////////////
//////////////
//
//   Use  : Use this function to get whole string starting
After
//          the Nth List separated by any of characters from
//          theSeparatorList.
//
////////////////////////////////////////////////////////////////
/////////////////
char * CUtility :: GetAfterNthList(char * str, char
*theSeparatorList, int thePosition)
{
char *aTransitionString;
char *anOutputString;
int length;
int difference;
length = strlen(str)+1;
anOutputString = (char *)xvt_malloc(length);
strcpy(anOutputString, str);

aTransitionString = NullTerminateUptoNthList(anOutputString,
theSeparatorList, thePosition);
difference = aTransitionString-anOutputString ;

if (aTransitionString)
   {
   length = strlen(aTransitionString);
   aTransitionString = NULL;
   aTransitionString = &str[(difference)+length+1] ;
     }
xvt_free (anOutputString);
return aTransitionString;
}
////////////////////////////////////////////////////////////////
//////////////////
//
//
```

```
//   Use  : Use this function to move through a str having
//
//          null characters to separate.
//
//
//
////////////////////////////////////////////////////////////
//////////////////
char * CUtility :: GetNthNullTerminatedList(char * str, int
thePosition)
{
char *aTransitionString;
int length;
int aCount;
length = strlen(str);
for (aCount = 1 , aTransitionString = str;
     (aCount < thePosition);
     aTransitionString += (length+1),aCount =
(length)?aCount+1:aCount)
       length = strlen(aTransitionString);
return aTransitionString;
}

////////////////////////////////////////////////////////////
//////////////////
//   Use      Use this function to replace the List at
thePosition
//        by theGivenString in str and store the result in
//        theBuffer
////////////////////////////////////////////////////////////
//////////////
char * CUtility :: ReplaceNthGivenList(char * str, char
*theSeparatorList,
            int thePosition, char *theGivenString, char
*theBuffer)
{
char *aTransitionString;
char *aLastString;
int numChars=0;
aLastString = GetNthList(str, theSeparatorList, thePosition,
&numChars);
aLastString += numChars;

strcpy(theBuffer, str);
```

```
aTransitionString = NullTerminateNthList(theBuffer,
theSeparatorList, thePosition);
if (!aTransitionString)
  aTransitionString = theBuffer;
strcpy(aTransitionString, theGivenString);
if (aLastString)
  strcat(theBuffer, aLastString);
return(theBuffer);
}
```

USER MANUAL

I. Minimum Requirements

A. Hardware

1. 80286 based PC-AT
2. 4MB RAM

B. Software

MS-Windows 3.1

II. Installation

Copy these files from floppy drive

A. to the directory of your choice

grapher.exe

B. to WINDOWS directory (default is C:\WINDOWS)

1. XWI322.DLL
2. XWI322TE.DLL

III. General

1. This is Grapher.
2. It maintains Graphs.
3. Each Graph is a set of Objects.
4. An Object can have a relation with any other object in a Graph.

IV. Menu Items

Information about the Menu Items

A. File

1. New

   Start a new Graph
2. Open

   Open an existing Graph
3. Close

   Close an active Graph
4. Save

   Save an active Graph
5. Save As

   Save the active Graph by another name
6. Exit - Exit the Grapher

B. Window

1. Tile

   Forms the active windows into a Tile display
2. Cascade

Forms the active windows into a Cascading display

3. Arrange Icons
   Arrnges the Icons in an order

C. Help

1. Contents
   Contents of help
2. About
   for the display of About Box

D. Object

1. Modify
   create/modify Objects
2. Destroy
   destroy Objects
3. Associate
   associate a function or an Executable file with an Object
4. Execute
   execute the process associated with the Object
5. Socialize
   set up relationships between various Objects.
   a) Plus Two more options to specify the relationships
   (1) Parent (default)
   (2) Child (default)

E. Options

1. Move
   select if you want to move the Objects around
2. Auto
   select if you want to autosize the Objects to be created/modified
3. Relation
   select if you want to view the relations as arrows
4. Node Information
   select if you want to view the information about Objects

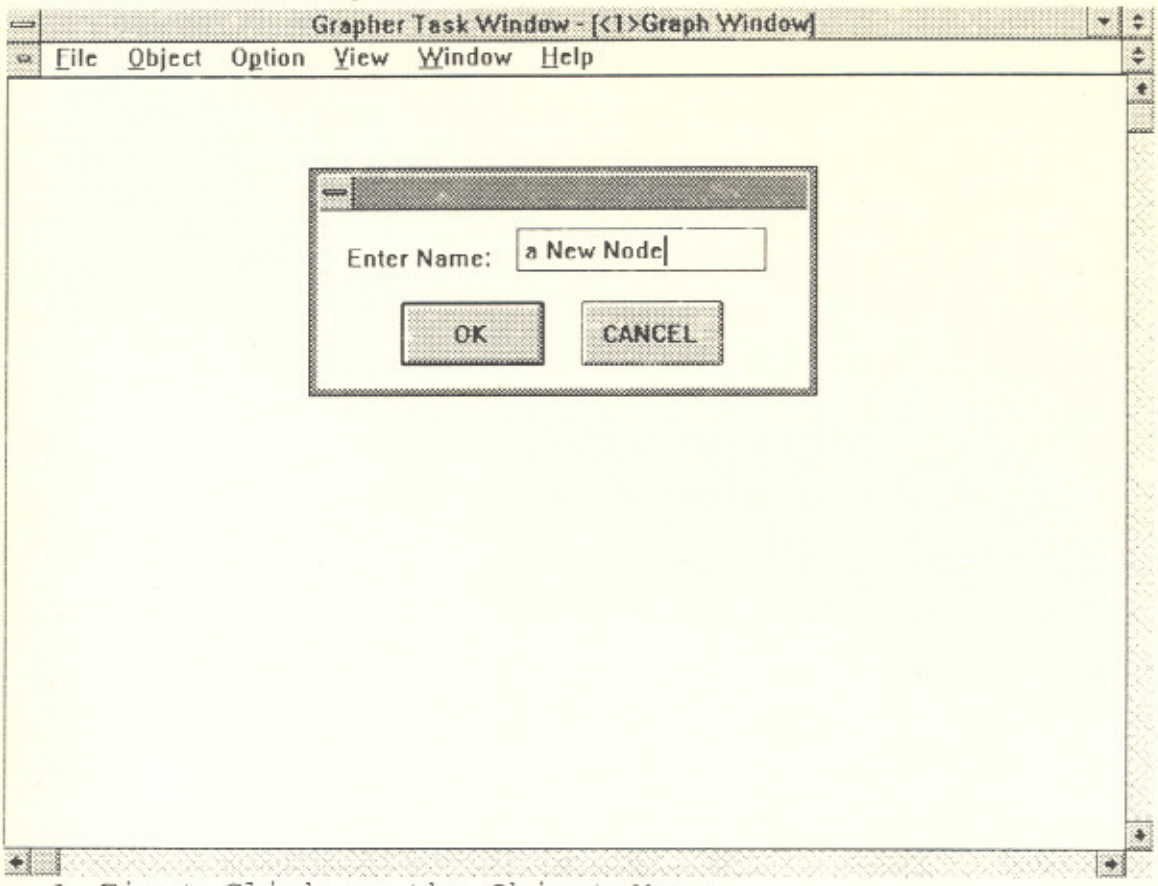
F. View

1. None
   No View of the Object
2. Button
   the Object is viewed as a button
3. Rectangle

the Object is viewed as a rectangle

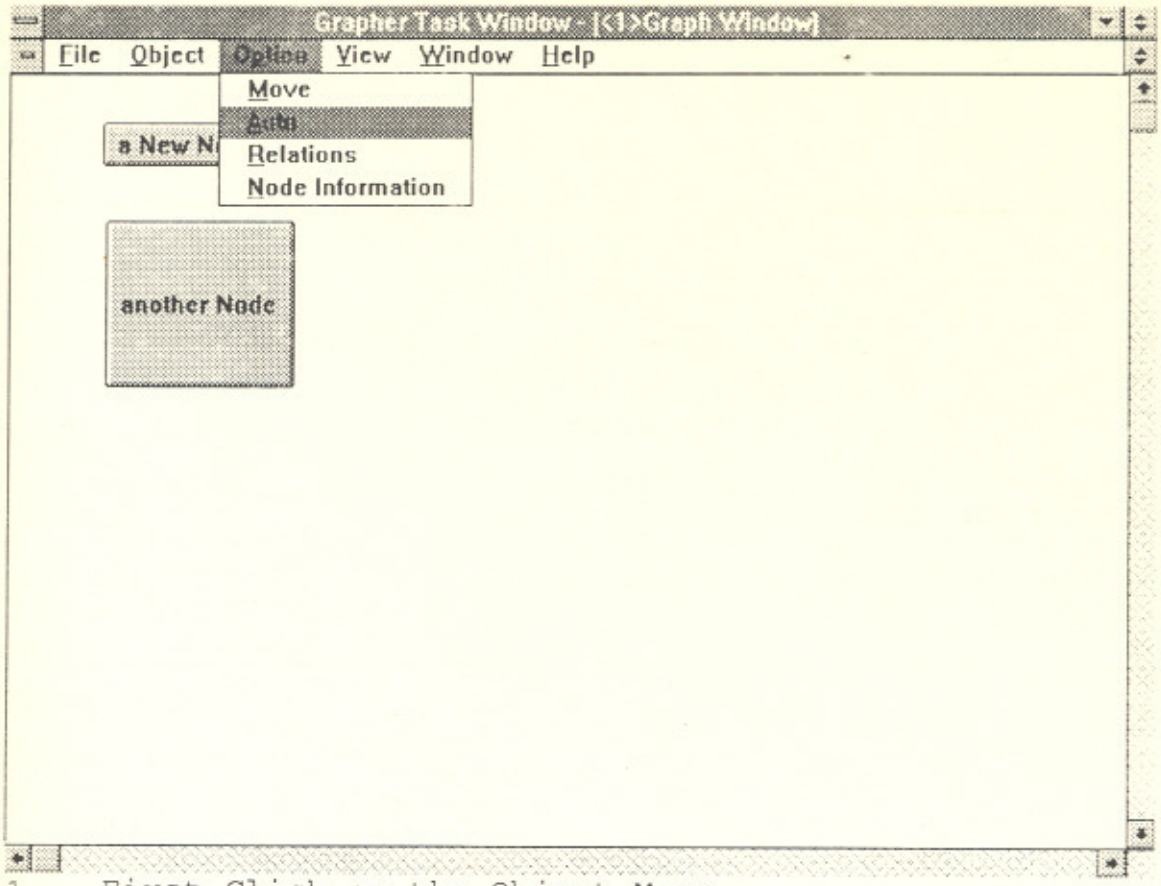
V. Some commonly asked questions

A. create an Object



1. First Click on the Object Menu
2. Here you select the Modify Item
3. Select the type of view to be associated to the Object from the View Menu
4. Place the mouse cursor at the place where you want your Object to be seen
5. Click with the Left Mouse button
6. After that a dialog box will appear asking the name of the Object
7. Fill the name and press Ok
8. If at that moment it is realised that Object is not needed select the Cancel button.

B. get user-defined Object size



1. First Click on the Object Menu.
2. Here you select the Modify Item.
3. After that Click on the Options Menu.
4. Set the Auto Option as unselected.
5. Place the mouse cursor at the place where you want your Object to be seen.
6. Click with the Left Mouse button and start dragging the mouse without releasing the button. A variable size rectangle will be formed as the mouse is moved. Size according to the need and release the button.
7. After that a dialog box will appear asking the name of the Object. Fill the name and press Ok.
8. If at that moment it is realised that Object is not needed select the Cancel button.

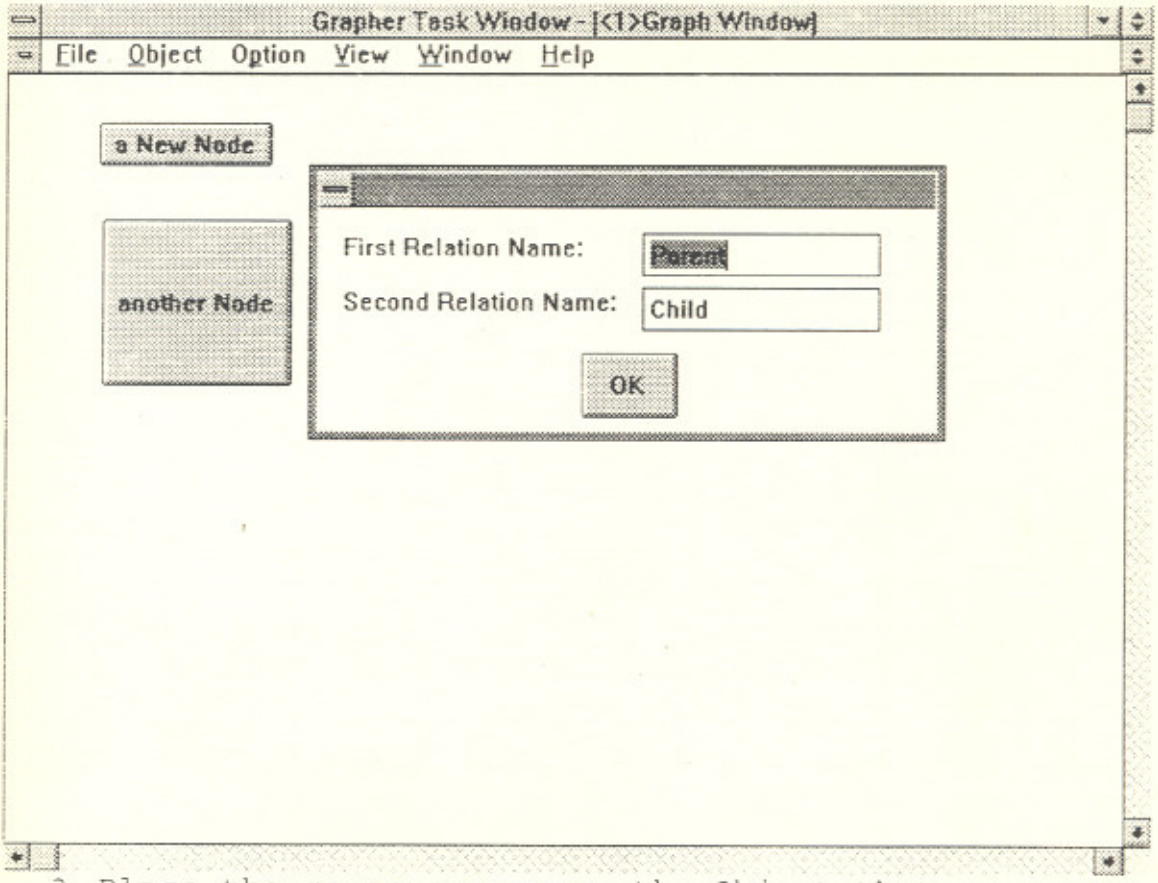
C. modify an Object

1. First Click on the Object Menu
2. Here you select the Modify Item

3. Place the mouse cursor on the Object that you want to be modified
4. Click with the Left Mouse button
5. After that a dialog box will appear showing the name of that Object. Modify the name and press Ok.
6. If at that moment it is realised that Object is not needed select the Cancel button.
7. "Remember" - If the Auto Option is checked at the time of modification then Objects with user defined size will be autosized.

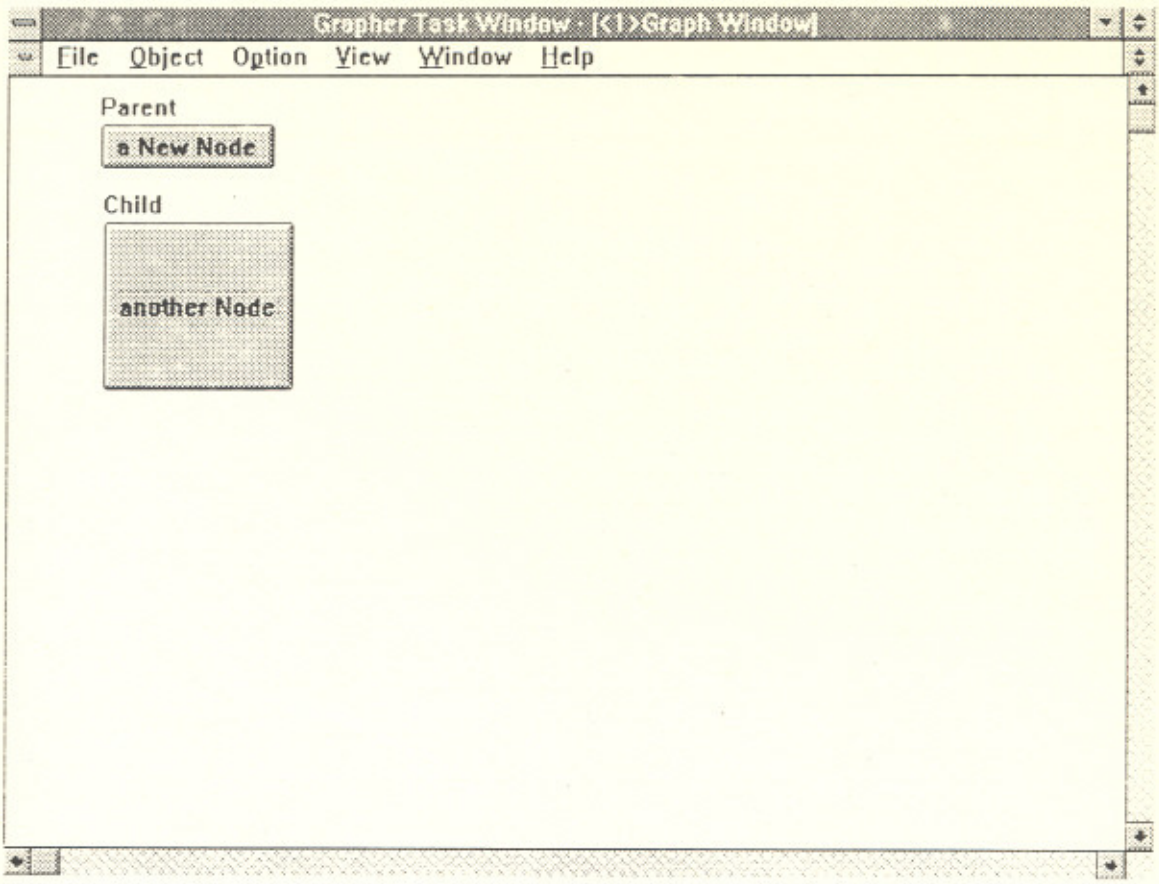
D. Set Up a relation

1. First Click on the Object Menu
2. First Click on the Socialize Menu Item



3. Place the mouse cursor on the Object that you want to related to some other Object. (If you are doing this for the first time in a New Graph, You will be prompted to enter the names of your choice signifying the relations that you want to form,

Actually a relation is between two Objects and both may give their own name to that relation)



4. The name of that relation will appear on the top left corner of that Object signifying that this Object is going to be related to some other Object, and that the other Object will be Viewing this Object reffered to by this name.
5. You can go on clicking in this way for one relation.
6. In case you want to add relations of other type then select that relation name from Object Menu.
7. If you want to deselect an Object designated to be socialized ,then click on it again.
8. Now when you chose any other Option from Object Menu these relations will be made permanent.

E. Associate a process

1. First Click on the Object Menu
2. First Click on the Associate Menu Item

3. After that click on the Object with which you want to associate a process
4. Choose Function or Executable to associate with it
5. In case of executable an exe can be associated, which can be executed in the Execute Mode.