Interstitial Brachytherapy Cancer Treatment Optimization Using Simulated Annealing and Artificial Neural Networks

By

Steven R. G. Miller

A Thesis Submitted to
The Faculty of Graduate Studies
In Partial Fulfilment of the Requirements
For the Degree of

Master of Science

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

Thesis Advisor: Professor W. Kinsner, Ph.D., P.Eng.

© Steven R. G. Miller, 2002

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

INTERSTITIAL BRACHYTHERAPY CANCER TREATMENT OPTIMIZATION USING SIMULATED ANNEALING AND ARTIFICIAL NEURAL NETWORKS

BY

STEVEN R.G. MILLER

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

STEVEN R.G. MILLER © 2002

Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film, and to University Microfilm Inc. to publish an abstract of this thesis/practicum.

The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

APPENDIX D

MANN SOURCE FILES

D.1 Back_Prop.cpp

```
// back_prop.cpp: implementation of the back_prop class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include <math.h>
#include "WinANN.h"
#include "back_prop.h"

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

#define TRAINING_FILE"training.dat"
#define WEIGHTS_FILE "weights.dat"
#define OUTPUT_FILE"output.dat"
#define TEST_FILE"test.dat"


//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

IMPLEMENT_DYNCREATE(back_prop,CWinThread)

back_prop::back_prop()
{

}

back_prop::~back_prop()
{

}


BOOL back_prop::InitInstance()
{
    m_NewData = FALSE;
    m_Done = FALSE;

    // call the main thread function
    go();
    return TRUE;
}

void back_prop::go()
{
    // Init the neural network here
float error_tolerance=0.1;
float total_error=0.0;
float avg_error_per_cycle=0.0;
float error_last_cycle=0.0;
float avgerr_per_pattern=0.0; // for the latest cycle
float error_last_pattern=0.0;
```

```
float learning_parameter=0.02;
unsigned startup=1;
long int vectors_in_buffer=0;
long int max_cycles=0;
long int patterns_per_cycle=0;
long int total_cycles, total_patterns=0;
int i;
FILE * training_file_ptr=NULL, * weights_file_ptr=NULL;
FILE * test_file_ptr=NULL, * data_file_ptr=NULL;
FILE * output_file_ptr=NULL, * p_MyOutputFile=NULL;
FILE * m_CoolingPlotFile=NULL;
int m_MoreTrain=0;
static long int m_Count=0;
CPoint OldPoint;


if ((output_file_ptr=fopen(OUTPUT_FILE,"w"))==NULL)
{
    AfxMessageBox("problem opening output file");
    exit(1);
}

// open my output file for writing
if ((p_MyOutputFile=fopen("MyOutput.dat","w"))==NULL)
{
    AfxMessageBox("problem opening my output file");
    exit(1);
}

// open cooling file for cooling profile
m_MoreTrain = 0;// default to no
if (m_TrainMode == 2)
{
    if ((m_CoolingPlotFile = fopen("CoolingPlot.dat","a")) == NULL)
    {
        AfxMessageBox("problem opening cooling plot file");
        exit(1);
    }
    // set the global flag for loading weights instead of random
    m_MoreTrain = 1;// mine to remember more training is required
    m_TrainMode = 1;// so that it treats all else as training
    CString s;
    s.Format("%d",m_Count);
    AfxMessageBox(s);
}
else if (m_TrainMode == 1)
{
    if ((m_CoolingPlotFile = fopen("CoolingPlot.dat","w")) == NULL)
    {
        AfxMessageBox("problem opening cooling plot file");
        exit(1);
    }
    m_Count=0;// total number of cycles
}


backp.set_training(m_TrainMode);


if (backp.get_training_value()==1)
{
    // ----------------------------------------
    //  Store values for the error_tolerance,
    //  and the learning_parameter
    // ----------------------------------------
    error_tolerance = m_ErrorLimit;
    learning_parameter = m_StartBeta;

    //----------------------------------------
    // open training file for reading
    //----------------------------------------
    if ((training_file_ptr=fopen(TRAINING_FILE,"r"))==NULL)
```

```
    {
        AfxMessageBox("Problem opening training file");
        exit(1);
    }

    data_file_ptr=training_file_ptr; // training on

    // Store the maximum number of cycles
    // each pass through the input data file is a cycle
    max_cycles = m_IterPerBeta;
}
else//not training
{
    if ((test_file_ptr=fopen(TEST_FILE,"r"))==NULL)
    {
        AfxMessageBox("problem opening test file");
        exit(1);
    }
    data_file_ptr=test_file_ptr; // training off
}



// get layer information
//backp.get_layer_info();//old way,dos/unix
//new way, get layer sizes from local variables as stored from main thread
int LayerSizes[5];
LayerSizes[0] = m_IL;
LayerSizes[1] = m_ML;
LayerSizes[2] = m_OL;
backp.set_layer_info(3,LayerSizes);// for thesis always use 3 layer

// set up the network connections
backp.set_up_network();

// initialize the weights if training to random numbers
if (backp.get_training_value()==1)
{
    if (!m_MoreTrain)// them just make random weights
    {
        // randomize weights for all layers; there is no
        // weight matrix associated with the input layer
        // weight file will be written after processing
        // so open for writing
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))==NULL)
        {
            AfxMessageBox("Problem opening weight file");
            exit(1);
        }
        backp.randomize_weights();
    }
    else//otherwise load in weights from the past
    {
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))==NULL)
        {
            AfxMessageBox("Problem opening weights file for reading");
            exit(1);
        }
        backp.read_weights(weights_file_ptr);
        fclose(weights_file_ptr);

        // now reopen weights file to be rewritten
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))==NULL)
        {
            AfxMessageBox("Problem opening weights file for writting");
            exit(1);
        }
    }//end else
}//end if training
else//backp.get_training_value()==0
{
    // read in the weight matrix defined by a
```

```
    // prior run of the backpropagation simulator
    // with training on
    if ((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))==NULL)
    {
        AfxMessageBox("Problem opening weights file for reading");
        exit(1);
    }
    backp.read_weights(weights_file_ptr);
}//end else



    // main loop

// training: continue looping until the total error is less than
//      the tolerance specified, or the maximum number of
//      cycles is exceeded; use both the forward signal propagation
//      and the backward error propagation phases. If the error
//      tolerance criteria is satisfied, save the weights in a file.
// no training: just proceed through the input data set once in the
//      forward signal propagation phase only. Read the starting
//      weights from a file.
// in both cases report the outputs on the screen

// if training is on, keep going through the input data
//      until the error is acceptable or the maximum number of cycles
//      is exceeded.
// if training is off, go through the input data once. report outputs
// with inputs to file output.dat


// intialize counters
    total_cycles=0; // a cycle is once through all the input data
    total_patterns=0; // a pattern is one entry in the input data
    startup=1;// first time through so that loops once at least
    vectors_in_buffer = MAX_VECTORS; // startup condition
    total_error = 0; //total error this cycle counter

    while ( ((backp.get_training_value()==1) && (avgerr_per_pattern> error_tolerance)
                && (total_cycles < max_cycles) && (vectors_in_buffer !=0))
            || ((backp.get_training_value()==0)&& (total_cycles < 1))
            || ((backp.get_training_value()==1)&& (startup==1)) )
    {
        if (startup)// only set variable once to save time
            startup=0;// no longer the first time through
        error_last_cycle=0; // reset for each cycle
        patterns_per_cycle=0; // reset for each cycle

        // process all the vectors in the datafile
        // going through one buffer at a time
        // pattern by pattern
        while ((vectors_in_buffer==MAX_VECTORS))
        {

            if (backp.get_training_value() == 0)
            {
                backp.set_training(1);
                vectors_in_buffer=backp.fill_IObuffer(data_file_ptr); // fill buffer
                backp.set_training(0);
            }
            else
                vectors_in_buffer=backp.fill_IObuffer(data_file_ptr); // fill buffer

            if (vectors_in_buffer < 0)
            {
                CString s;
                s.Format("error reading in vectors.....ABORTING\ncheck that there are extra linefeeds in
your data file,\nand that the number of layers and size of layers match the\nthe parameters provided.");
                AfxMessageBox(s);
                exit(1);
            }
```

```
            // process vectors
            for (i=0; i<vectors_in_buffer; i++)
            {
                // get next pattern
                if (backp.get_training_value() == 0)
                {
                    backp.set_training(1);
                    backp.set_up_pattern(i);
                    backp.set_training(0);
                }
                else
                    backp.set_up_pattern(i);

                total_patterns++;
                patterns_per_cycle++;

                // forward propagate
                backp.forward_prop();

                if (backp.get_training_value()==0)// store all of the vector output to a file
                {
                    backp.write_outputs(output_file_ptr,p_MyOutputFile);
                    backp.backward_prop(error_last_pattern);
                    error_last_cycle += error_last_pattern*error_last_pattern;
                }

                // back_propagate, if appropriate
                if (backp.get_training_value()==1)
                {
                    backp.backward_prop(error_last_pattern);
                    error_last_cycle += error_last_pattern*error_last_pattern;
                    backp.update_weights(learning_parameter);
                    // backp.list_weights(); // can see change in weights by using list_weights before
and after back_propagation
                }//end if
            }//end for
            error_last_pattern = 0;
        }//end while vectors_in_buffer

        //calculate the average error
        avgerr_per_pattern=((float)sqrt((double)error_last_cycle/patterns_per_cycle));
        total_error += error_last_cycle;
        total_cycles++;
        m_Count++;

        //Added 21/4/2000
        CString s;
        s.Format("Error on test set is: %f",avgerr_per_pattern);
        if (backp.get_training_value() == 0)
            AfxMessageBox(s);



        if (backp.get_training_value()==1)
            fprintf(m_CoolingPlotFile,"%f\n",avgerr_per_pattern);

        // refresh the display:
        m_CurrBeta = learning_parameter;
        if (avgerr_per_pattern > 0.0001)
            m_CurrError = avgerr_per_pattern;
        else
            m_CurrError = 0.001;
        m_CurrIteration = total_cycles;

        CPoint NewPoint;
        NewPoint.x = m_Canvas.left + (m_Count-1) * ((double)(m_Canvas.right-m_Canvas.left)/(dou-
ble)(m_MaxIterations-1));
        NewPoint.y = m_Canvas.bottom - avgerr_per_pattern * ((m_Canvas.bottom-m_Canvas.top)/
m_MaxError);
        if (NewPoint.y < m_Canvas.top)
            NewPoint.y = m_Canvas.top+5;
```

```
        if ((m_Count > 1) && (backp.get_training_value()==1))
        {
            if ((NewPoint.x != GetPoint(NumPoints()-1).x) && (NewPoint.y != GetPoint(NumPoints()-1).y))
            {
                AddPoint(NewPoint);
                m_NewData = TRUE;
                m_DlgWnd->RedrawWindow();
            }
        }
        else if (backp.get_training_value()==1)
        {
            AddPoint(NewPoint);
            m_NewData = TRUE;
            m_DlgWnd->RedrawWindow();
        }

        if ((total_cycles == max_cycles) && (learning_parameter>0.01))
        {
            learning_parameter *= 0.75;
            total_cycles = 0;
        }

        fseek(data_file_ptr, 0L, SEEK_SET); // reset the file pointer
                                            // to the beginning of
                                            // the file
        vectors_in_buffer = MAX_VECTORS; // reset

    } // end main loop


//  cout << "-----------------------------------------------------\n";
//  cout << " done:results in file output.dat\n";
//  cout << "training: last vector only\n";
//  cout << "not training: full cycle\n\n";
    if (backp.get_training_value()==1)
    {
        backp.write_weights(weights_file_ptr);
        backp.write_outputs(output_file_ptr,p_MyOutputFile);
        avg_error_per_cycle= (float)sqrt((double)total_error/total_cycles);
        error_last_cycle=(float)sqrt((double)error_last_cycle);

//  cout << "weights saved in file weights.dat\n";
//  cout << "\n";
//  cout << "---->average error per cycle = " << avg_error_per_cycle << " <---\n";
//  cout << "---->error last cycle = " << error_last_cycle << " <---\n";
//  cout << "->error last cycle per pattern= " << avgerr_per_pattern << " <---\n";

    }

//  cout << "------------>total cycles = " << total_cycles << " <---\n";
//  cout << "------------>total patterns = " << total_patterns << " <---\n";
//  cout << "-----------------------------------------------------\n";

    // close all files
    fclose(data_file_ptr);
    fclose(weights_file_ptr);
    fclose(p_MyOutputFile);
    fclose(output_file_ptr);
    if (m_CoolingPlotFile)
        fclose(m_CoolingPlotFile);
    ////////////////////////////////

    m_Done = TRUE;
    m_DlgWnd->RedrawWindow();
}

int back_prop::NumPoints()
{
    return m_Data.GetSize();
}
```

```
CPoint back_prop::GetPoint(int index)
{
    return m_Data[index];
}

void back_prop::AddPoint(CPoint point)
{
    m_Data.Add(point);
}
```

D.2 Back_Prop.h

```
// back_prop.h: interface for the back_prop class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_BACK_PROP_H__06F17D51_F143_11D3_8089_0090276F4183__INCLUDED_)
#define AFX_BACK_PROP_H__06F17D51_F143_11D3_8089_0090276F4183__INCLUDED_

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

#include "layer.h"

class back_prop  : public CWinThread
{
    DECLARE_DYNCREATE(back_prop)
public:
    int m_OldMaxIterations;
    BOOL m_Done;
    BOOL m_NewData;
    back_prop();
    virtual ~back_prop();
    virtual BOOL InitInstance();
    void go();
    network backp;
    floatm_ErrorLimit;
    int     m_IterPerBeta;
    int     m_IL;
    int     m_ML;
    int     m_OL;
    floatm_StartBeta;
    floatm_StopBeta;
    floatm_CurrBeta;
    floatm_CurrError;
    int     m_MaxIterations;
    floatm_MaxError;
    int     m_TrainMode;
    int     m_CurrIteration;
    CRect m_Canvas;
    CWnd * m_DlgWnd;
    void AddPoint(CPoint point);
    CPoint GetPoint(int index);
    int NumPoints();
    CArray<CPoint,CPoint> m_Data;
};

#endif // !defined(AFX_BACK_PROP_H__06F17D51_F143_11D3_8089_0090276F4183__INCLUDED_)
```

D.3 ConfigDlg.cpp

```
// ConfigDlg.cpp : implementation file
//

#include "stdafx.h"
#include "WinANN.h"
#include "ConfigDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CConfigDlg dialog


CConfigDlg::CConfigDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CConfigDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CConfigDlg)
    m_ErrorLimit = 0.0f;
    m_IterPerBeta = 0;
    m_ML = 0;
    m_OL = 0;
    m_StartBeta = 0.0f;
    m_IL = 0;
    m_StopBeta = 0.0f;
    m_TrainMode = 0;
    //}}AFX_DATA_INIT

}


void CConfigDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CConfigDlg)
    DDX_Control(pDX, IDC_TRAINMODE, m_TrainModeBox);
    DDX_Control(pDX, IDC_OL, m_OLBox);
    DDX_Control(pDX, IDC_ML, m_MLBox);
    DDX_Control(pDX, IDC_IL, m_ILBox);
    DDX_Text(pDX, IDC_ERRORLIMIT, m_ErrorLimit);
    DDV_MinMaxFloat(pDX, m_ErrorLimit, 1.e-003f, 100.f);
    DDX_Text(pDX, IDC_ITPERBETA, m_IterPerBeta);
    DDV_MinMaxInt(pDX, m_IterPerBeta, 1, 10000);
    DDX_Text(pDX, IDC_ML, m_ML);
    DDV_MinMaxInt(pDX, m_ML, 2, 100);
    DDX_Text(pDX, IDC_OL, m_OL);
    DDV_MinMaxInt(pDX, m_OL, 1, 10000);
    DDX_Text(pDX, IDC_STARTBETA, m_StartBeta);
    DDV_MinMaxFloat(pDX, m_StartBeta, 1.e-003f, 1.f);
    DDX_Text(pDX, IDC_IL, m_IL);
    DDV_MinMaxInt(pDX, m_IL, 1, 10000);
    DDX_Text(pDX, IDC_STOPBETA, m_StopBeta);
    DDV_MinMaxFloat(pDX, m_StopBeta, 1.e-002f, 1.f);
    DDX_Text(pDX, IDC_TRAINMODE, m_TrainMode);
    DDV_MinMaxInt(pDX, m_TrainMode, 0, 2);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CConfigDlg, CDialog)
    //{{AFX_MSG_MAP(CConfigDlg)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CConfigDlg message handlers
```

```
BOOL CConfigDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    if (m_TrainMore == TRUE)
    {
        m_ILBox.EnableWindow(FALSE);
        m_MLBox.EnableWindow(FALSE);
        m_OLBox.EnableWindow(FALSE);
        m_TrainModeBox.EnableWindow(FALSE);
    }

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

D.4 ConfigDlg.h

```
#if !defined(AFX_CONFIGDLG_H__0E253D0D_EFB8_11D3_8089_0090276F4183__INCLUDED_)
#define AFX_CONFIGDLG_H__0E253D0D_EFB8_11D3_8089_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ConfigDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CConfigDlg dialog

class CConfigDlg : public CDialog
{
// Construction
public:
    CConfigDlg(CWnd* pParent = NULL);   // standard constructor
    BOOL m_TrainMore;//set to true if this is the train more function calling it.

// Dialog Data
    //{{AFX_DATA(CConfigDlg)
    enum { IDD = IDD_CONFIG_DIALOG };
    CEditm_TrainModeBox;
    CEditm_OLBox;
    CEditm_MLBox;
    CEditm_ILBox;
    floatm_ErrorLimit;
    int     m_IterPerBeta;
    int     m_ML;
    int     m_OL;
    floatm_StartBeta;
    int     m_IL;
    floatm_StopBeta;
    int     m_TrainMode;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CConfigDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CConfigDlg)
```

```
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_CONFIGDLG_H__0E253D0D_EFB8_11D3_8089_0090276F4183__INCLUDED_)
```

D.5 Layer.cpp

```
// layer.cpp
// compile for floating point hardware if available
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "layer.h"


inline float squash(float input)
// squashing function
// use sigmoid -- can customize to something
// else if desired; can add a bias term too
//
{
if (input < -50)
    return 0.0;
elseif (input > 50)
        return 1.0;
    else return (float)(1/(1+exp(-(double)input)));

}


inline float randomweight(unsigned init)
{
int num;
// random number generator
// will return a floating point
// value between -1 and 1

if (init==1)// seed the generator
    srand ((unsigned)time(NULL));

num=rand() % 100;

return 2*(float(num/100.00))-1;
}

// the next function is needed for Turbo C++
// and Borland C++ to link in the appropriate
// functions for fscanf floating point formats:
static void force_fpf()
{
    float x, *y;
    y=&x;
    x=*y;
}

// ----------------------------------------
//             input layer
//----------------------------------------
input_layer::input_layer(int i, int o)
{
```

```
num_inputs=i;
num_outputs=o;

outputs = new float[num_outputs];
if (outputs==0)
    {
    cout << "not enough memory\n";
    cout << "choose a smaller architecture\n";
    exit(1);
    }
}

input_layer::~input_layer()
{
delete [num_outputs] outputs;
}


void input_layer::calc_out()
{
//nothing to do, yet
}



// ----------------------------------------
//             output layer
//----------------------------------------




output_layer::output_layer(int i, int o)
{

num_inputs=i;
num_outputs=o;
weights = new float[num_inputs*num_outputs];
output_errors = new float[num_outputs];
back_errors = new float[num_inputs];
outputs = new float[num_outputs];
expected_values = new float[num_outputs];
if ((weights==0)||(output_errors==0)||(back_errors==0)
    ||(outputs==0)||(expected_values==0))
    {
    cout << "not enough memory\n";
    cout << "choose a smaller architecture\n";
    exit(1);
    }

}

output_layer::~output_layer()
{
// some compilers may require the array
// size in the delete statement; those
// conforming to Ansi C++ will not
delete [num_outputs*num_inputs] weights;
delete [num_outputs] output_errors;
delete [num_inputs] back_errors;
delete [num_outputs] outputs;

}


void output_layer::calc_out()
{
```

```
int i,j,k;
float accumulator=0.0;


for (j=0; j<num_outputs; j++)
    {

    for (i=0; i<num_inputs; i++)

        {
        k=i*num_outputs;
        if (weights[k+j]*weights[k+j] > 1000000.0)
            {
            cout << "weights are blowing up\n";
            cout << "try a smaller learning constant\n";
            cout << "e.g. beta=0.02    aborting...\n";
            exit(1);
            }
        outputs[j]=weights[k+j]*(*(inputs+i));
        accumulator+=outputs[j];
        }
    // use the sigmoid squash function
    outputs[j]=squash(accumulator);
    accumulator=0;
    }

}



void output_layer::calc_error(float & error)
{
int i, j, k;
float accumulator=0;
float total_error=0;

for (j=0; j<num_outputs; j++)
    {
    output_errors[j] = expected_values[j]-outputs[j];
    total_error+=output_errors[j];
    }

error=total_error;

for (i=0; i<num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j<num_outputs; j++)
        {
        back_errors[i]=
            weights[k+j]*output_errors[j];
        accumulator+=back_errors[i];
        }
    back_errors[i]=accumulator;
    accumulator=0;
    // now multiply by derivative of
    // sigmoid squashing function, which is
    // just the input*(1-input)
    back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
    }

}

void output_layer::randomize_weights()
{
int i, j, k;
const unsigned first_time=1;

const unsigned not_first_time=0;
float discard;

discard=randomweight(first_time);
```

```
for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        weights[k+j]=randomweight(not_first_time);
    }
}

void output_layer::update_weights(const float beta)
{
int i, j, k;

// learning law: weight_change =
//      beta*output_error*input

for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        weights[k+j] +=
            beta*output_errors[j]*(*(inputs+i));
    }

}

void output_layer::list_weights()
{
int i, j, k;

for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        cout << "weight["<<i<<","<<
            j<<"] is: "<<weights[k+j];
    }

}

void output_layer::list_errors()
{
int i, j;

for (i=0; i< num_inputs; i++)
    cout << "backerror["<<i<<
        "] is : "<<back_errors[i]<<"\n";

for (j=0; j< num_outputs; j++)
    cout << "outputerrors["<<j<<
            "] is: "<<output_errors[j]<<"\n";

}


void output_layer::write_weights(int layer_no,
        FILE * weights_file_ptr)
{
int i, j, k;

// assume file is already open and ready for
// writing

// prepend the layer_no to all lines of data
// format:
//      layer_noweight[0,0] weight[0,1] ...
//      layer_noweight[1,0] weight[1,1] ...
//      ...

for (i=0; i< num_inputs; i++)
    {
    fprintf(weights_file_ptr,"%i ",layer_no);
```

```
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
    {
    fprintf(weights_file_ptr,"%f ",
            weights[k+j]);
    }
    fprintf(weights_file_ptr,"\n");
    }


}

void output_layer::read_weights(int layer_no,
        FILE * weights_file_ptr)
{
int i, j, k;


// assume file is already open and ready for
// reading

// look for the prepended layer_no
// format:
//      layer_noweight[0,0] weight[0,1] ...
//      layer_noweight[1,0] weight[1,1] ...
//      ...
while (1)

    {

    fscanf(weights_file_ptr,"%i",&j);
    if ((j==layer_no)|| (feof(weights_file_ptr)))
        break;
    else
        {
        while (fgetc(weights_file_ptr) != '\n')
            {;}// get rest of line
        }
    }

if (!(feof(weights_file_ptr)))
    {
    // continue getting first line
    i=0;
    for (j=0; j< num_outputs; j++)
            {

            fscanf(weights_file_ptr,"%f",
                    &weights[j]); // i*num_outputs = 0
        }
    fscanf(weights_file_ptr,"\n");



    // now get the other lines
    for (i=1; i< num_inputs; i++)
        {
        fscanf(weights_file_ptr,"%i",&layer_no);
        k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        {
        fscanf(weights_file_ptr,"%f",
            &weights[k+j]);
             }

    }
    fscanf(weights_file_ptr,"\n");
    }


else cout << "end of file reached\n";
```

```
}
void output_layer::list_outputs()
{
int j;

for (j=0; j< num_outputs; j++)
    {
    cout << "outputs["<<j
        <<"] is: "<<outputs[j]<<"\n";
    }

}


// ----------------------------------------
//             middle layer
//----------------------------------------


middle_layer::middle_layer(int i, int o):
    output_layer(i,o)
{

}

middle_layer::~middle_layer()
{
    delete [num_outputs*num_inputs] weights;
    delete [num_outputs] output_errors;
    delete [num_inputs] back_errors;
    delete [num_outputs] outputs;
}//end middle layer


void middle_layer::calc_error()
{
int i, j, k;
float accumulator=0;

for (i=0; i<num_inputs; i++)
{
    k=i*num_outputs;
    for (j=0; j<num_outputs; j++)
    {
        back_errors[i]=weights[k+j]*(*(output_errors+j));
        accumulator+=back_errors[i];
    }
    back_errors[i]=accumulator;
    accumulator=0;
    // now multiply by derivative of
    // sigmoid squashing function, which is
    // just the input*(1-input)
    back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
}
}//end calc_error


// ----------------------------------------
//             network stuff
//----------------------------------------


network::network()
{
position=0L;
}

network::~network()
{
int i,j,k;
```

```
i=layer_ptr[0]->num_outputs;// inputs
j=layer_ptr[number_of_layers-1]->num_outputs; //outputs
k=MAX_VECTORS;


delete [(i+j)*k]buffer;
}

void network::set_training(const unsigned & value)
{
training=value;
}

unsigned network::get_training_value()
{
return training;
}


void network::get_layer_info()
{
int i;

//-----------------------------------------
//
//  Get layer sizes for the network
//
// ----------------------------------------



cout << " Please enter in the number of layers for your network.\n";
cout << " You can have a minimum of 3 to a maximum of 5. \n";
cout << " 3 implies 1 hidden layer; 5 implies 3 hidden layers : \n\n";

cin >> number_of_layers;

cout << " Enter in the layer sizes separated by spaces.\n";
cout << " For a network with 3 neurons in the input layer,\n";
cout << " 2 neurons in a hidden layer, and 4 neurons in the\n";
cout << " output layer, you would enter: 3 2 4 .\n";
cout << " You can have up to 3 hidden layers,for five maximum entries :\n\n";

for (i=0; i<number_of_layers; i++)
    {
    cin >> layer_size[i];
    }

// -----------------------------------------------------
// size of layers:
//      input_layer layer_size[0]
//      output_layerlayer_size[number_of_layers-1]
//      middle_layerslayer_size[1]
//                  optional: layer_size[number_of_layers-3]
//                  optional: layer_size[number_of_layers-2]
//-----------------------------------------------------


}


void network::set_layer_info(int NumLayer,int LayerSize[MAX_LAYERS])
{
    number_of_layers = NumLayer;
    for (int i=0; i<NumLayer; i++)
    {
        layer_size[i] = LayerSize[i];
    }
}

void network::set_up_network()
{
```

```
int i,j,k;
//--------------------------------------------------------
// Construct the layers
//
//--------------------------------------------------------



layer_ptr[0] = new input_layer(0,layer_size[0]);

for (i=0;i<(number_of_layers-1);i++)
    {
    layer_ptr[i+1] =
    new middle_layer(layer_size[i],layer_size[i+1]);
    }

layer_ptr[number_of_layers-1] = new
output_layer(layer_size[number_of_layers-2],layer_size[number_of_layers-1]);

for (i=0;i<(number_of_layers-1);i++)
    {
    if (layer_ptr[i] == 0)
        {
        cout << "insufficient memory\n";
        cout << "use a smaller architecture\n";
        exit(1);
        }
    }

//--------------------------------------------------------
// Connect the layers
//
//--------------------------------------------------------
// set inputs to previous layer outputs for all layers,
//  except the input layer

for (i=1; i< number_of_layers; i++)
    layer_ptr[i]->inputs = layer_ptr[i-1]->outputs;



// for back_propagation, set output_errors to next layer
//      back_errors for all layers except the output
//      layer and input layer


for (i=1; i< number_of_layers -1; i++)
    ((output_layer *)layer_ptr[i])->output_errors =
        ((output_layer *)layer_ptr[i+1])->back_errors;

// define the IObuffer that caches data from
// the datafile
i=layer_ptr[0]->num_outputs;// inputs
j=layer_ptr[number_of_layers-1]->num_outputs; //outputs
k=MAX_VECTORS;

buffer=new
    float[(i+j)*k];
if (buffer==0)
    {
    cout << "insufficient memory for buffer\n";
    exit(1);
    }
}

void network::randomize_weights()
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->randomize_weights();
```

```
}


void network::update_weights(const float beta)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])->update_weights(beta);
}


void network::write_weights(FILE * weights_file_ptr)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->write_weights(i,weights_file_ptr);
}


void network::read_weights(FILE * weights_file_ptr)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->read_weights(i,weights_file_ptr);
}


void network::list_weights()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_weights();
    }
}

void network::list_outputs()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_outputs();
    }
}

void network::write_outputs(FILE *outfile,FILE * m_OutputFile)
{
int i, ins, outs;
ins=layer_ptr[0]->num_outputs;
outs=layer_ptr[number_of_layers-1]->num_outputs;
float temp;

//MINE ADDED Sept. 9, 1999
int m_Count=0;
float MinOutputAtSourceLocation = 1.2f;
///////DONE MINE

fprintf(outfile,"for input vector:\n");

for (i=0; i<ins; i++)
    {
```

```
    temp=layer_ptr[0]->outputs[i];
    fprintf(outfile,"%.5f  ",temp);
    // MY ADDED CODE Sept. 9,1999
    if (temp == 1.0)// assume this is a source
    {
        m_Count++;
        if (layer_ptr[number_of_layers-1]->outputs[i]<MinOutputAtSourceLocation)
            MinOutputAtSourceLocation = layer_ptr[number_of_layers-1]->outputs[i];
    }
    ////////////// DONE MINE
    }


fprintf(outfile,"\noutput vector is (%d, %f):\n",m_Count,MinOutputAtSourceLocation);

for (i=0; i<outs; i++)
    {
    temp=layer_ptr[number_of_layers-1]->
    outputs[i];
    fprintf(outfile,"%.5f  ",temp);

    }

/////MINE ADDED Sept. 9, 1999
fprintf(outfile,"\nMy output vector is:\n");

for (i=0; i<outs; i++)
    {
    temp=layer_ptr[number_of_layers-1]->
    outputs[i];
    if (temp>MinOutputAtSourceLocation)
        fprintf(outfile,"1.0 ");
    else
        fprintf(outfile,"0.0 ");
    }
int count=0;
for (i=0; i<outs; i++)
{
    temp=layer_ptr[number_of_layers-1]->outputs[i];

    if (count==30)
    {
        count=0;
        fprintf(m_OutputFile,"\n");
    }
    fprintf(m_OutputFile,"%f ",temp);
    count++;
}
    fprintf(m_OutputFile,"\n\n");
/////////DONE MINE



if (training==1)
{
fprintf(outfile,"\nexpected output vector is:\n");

for (i=0; i<outs; i++)
    {
    temp=((output_layer *)(layer_ptr[number_of_layers-1]))->
    expected_values[i];
    fprintf(outfile,"%f  ",temp);

    }
}

fprintf(outfile,"\n----------------------\n");

}
```

```
void network::list_errors()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_errors();
    }
}



int network::fill_IObuffer(FILE * inputfile)
{
// this routine fills memory with
// an array of input, output vectors
// up to a maximum capacity of
// MAX_INPUT_VECTORS_IN_ARRAY
// the return value is the number of read
// vectors


int i, k, count, veclength;

int ins, outs;

ins=layer_ptr[0]->num_outputs;

outs=layer_ptr[number_of_layers-1]->num_outputs;

if (training==1)
    veclength=ins+outs;
else
    veclength=ins;

count=0;
while  ((count<MAX_VECTORS)&&
        (!feof(inputfile)))
    {
    k=count*(veclength);
    for (i=0; i<veclength; i++)
        {
        fscanf(inputfile,"%f",&buffer[k+i]);
        }
    fscanf(inputfile,"\n");
    count++;
    }

if (!(ferror(inputfile)))
    return count;
else return -1; // error condition

}


void network::set_up_pattern(int buffer_index)
{
// read one vector into the network
int i, k;
int ins, outs;

ins=layer_ptr[0]->num_outputs;
outs=layer_ptr[number_of_layers-1]->num_outputs;
if (training==1)
    k=buffer_index*(ins+outs);
else
    k=buffer_index*ins;

for (i=0; i<ins; i++)
```

```
    layer_ptr[0]->outputs[i]=buffer[k+i];

if (training==1)
{
    for (i=0; i<outs; i++)

        ((output_layer *)layer_ptr[number_of_layers-1])->
            expected_values[i]=buffer[k+i+ins];
}

}



void network::forward_prop()
{
int i;
for (i=0; i<number_of_layers; i++)
    {
    layer_ptr[i]->calc_out(); //polymorphic
                // function
    }
}

void network::backward_prop(float & toterror)
{
int i;

// error for the output layer
((output_layer*)layer_ptr[number_of_layers-1])->
            calc_error(toterror);

// error for the middle layer(s)
for (i=number_of_layers-2; i>0; i--)
    {
    ((middle_layer*)layer_ptr[i])->
            calc_error();

    }

}
```

D.6 Layer.h

```
// layer.h
// header file for the layer class heirarchy and
// the network class

#define MAX_LAYERS5
#define MAX_VECTORS100


class network;

class layer
{

protected:

    int num_inputs;
    int num_outputs;
    float *outputs;// pointer to array of outputs
    float *inputs; // pointer to array of inputs, which
                   // are outputs of some other layer

    friend network;

public:
```

```
    virtual void calc_out()=0;
};


class input_layer: public layer
{

private:


public:

    input_layer(int, int);
    ~input_layer();
    virtual void calc_out();

};

class middle_layer;

class output_layer:public layer
{
protected:

    float * weights;
    float * output_errors; // array of errors at output
    float * back_errors; // array of errors back-propagated
    float * expected_values;// to inputs
    friend network;

public:


    output_layer(int, int);
    ~output_layer();
    virtual void calc_out();
    void calc_error(float &);
    void randomize_weights();
    void update_weights(const float);
    void list_weights();
    void write_weights(int, FILE *);
    void read_weights(int, FILE *);
    void list_errors();
    void list_outputs();
};



class middle_layer:public output_layer
{

private:

public:
    middle_layer(int, int);
    ~middle_layer();
    void calc_error();
};


class network

{

private:

layer *layer_ptr[MAX_LAYERS];
    int number_of_layers;
    int layer_size[MAX_LAYERS];
```

```
    float *buffer;
    fpos_t position;
    unsigned training;

public:
    network();
    ~network();
    void set_training(const unsigned &);
    unsigned get_training_value();
    void get_layer_info();
    void set_layer_info(int NumLayer,int LayerSize[MAX_LAYERS]);
    void set_up_network();
    void randomize_weights();
    void update_weights(const float);
    void write_weights(FILE *);
    void read_weights(FILE *);
    void list_weights();
    void write_outputs(FILE *,FILE *);
    void list_outputs();
    void list_errors();
    void forward_prop();
    void backward_prop(float &);
    int fill_IObuffer(FILE *);
    void set_up_pattern(int);

};
```

D.7 Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by WinANN.rc
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_WINANN_DIALOG               102
#define IDR_MAINFRAME                   128
#define IDB_TITLE                       129
#define IDD_CONFIG_DIALOG               130
#define IDC_ERRORLIMIT                  1000
#define IDC_STARTBETA                   1001
#define IDC_ITPERBETA                   1002
#define IDC_IL                          1003
#define IDC_ML                          1004
#define IDC_OL                          1005
#define IDC_STOPBETA                    1006
#define IDC_CURRBETA                    1007
#define IDC_TRAINMODE                   1007
#define IDC_CURRERROR                   1008
#define IDC_MAXITER                     1009
#define IDC_MAXERROR                    1010
#define IDC_TRAINING                    1011
#define IDC_GO                          1012
#define IDC_GRAPH                       1013
#define IDC_CURRITER                    1014
#define IDC_LOGO                        1015
#define IDC_PAUSE                       1016
#define IDC_TEST                        1017
#define IDC_TRAINMORE                   1018

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        131
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1019
```

```
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

D.8 WinANN.cpp

```
// WinANN.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include <math.h>
#include "WinANN.h"
#include "WinANNDlg.h"
#include "ConfigDlg.h"



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

/////////////////////////////////////////////////////////////////////////////
// CWinANNApp

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

/////////////////////////////////////////////////////////////////////////////
// CWinANNApp construction

CWinANNApp::CWinANNApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CWinANNApp object

CWinANNApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CWinANNApp initialization

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

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

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

    //get the configuration information
```

```
    CConfigDlg Cdlg;
    Cdlg.m_ErrorLimit = 0.1;
    Cdlg.m_IL = 901;
    Cdlg.m_IterPerBeta = 500;
    Cdlg.m_ML = 12;
    Cdlg.m_OL = 900;
    Cdlg.m_StartBeta = 0.75;
    Cdlg.m_StopBeta = 0.01;
    Cdlg.m_TrainMode = 1;
    Cdlg.m_TrainMore = FALSE;

    if (Cdlg.DoModal() == IDOK)
    {

        CWinANNDlg dlg;
        m_pMainWnd = &dlg;
        //set static text tags:
        dlg.m_ErrorLimit = Cdlg.m_ErrorLimit;
        dlg.m_IL = Cdlg.m_IL;
        dlg.m_IterPerBeta = Cdlg.m_IterPerBeta;
        dlg.m_ML = Cdlg.m_ML;
        dlg.m_OL = Cdlg.m_OL;
        dlg.m_StartBeta = Cdlg.m_StartBeta;
        dlg.m_StopBeta = Cdlg.m_StopBeta;
        dlg.m_TrainMode = Cdlg.m_TrainMode;

        //calculate the x size of the canvas graph
        double TempMI=0.0f;
        TempMI = log((double)dlg.m_StopBeta/(double)dlg.m_StartBeta);
        TempMI = TempMI/log(0.75);
        TempMI += 1.5f;
        TempMI *= 1.1;
        TempMI = floor(TempMI);
        TempMI *= dlg.m_IterPerBeta;
        dlg.m_MaxIterations = TempMI;

        dlg.m_CurrBeta = dlg.m_StartBeta;
        dlg.m_CurrError = 0.001;
        dlg.m_MaxError = 50.0;
        dlg.m_CurrIteration = 0;


        int nResponse = dlg.DoModal();


        if (nResponse == IDOK)
        {
            // TODO: Place code here to handle when the dialog is
            //  dismissed with OK
        }
        else if (nResponse == IDCANCEL)
        {
            // TODO: Place code here to handle when the dialog is
            //  dismissed with Cancel
        }
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    //  application, rather than start the application's message pump.
    return FALSE;
}
```

D.9 WinANN.h

```
// WinANN.h : main header file for the WINANN application
//

#if !defined(AFX_WINANN_H__0E253D02_EFB8_11D3_8089_0090276F4183__INCLUDED_)
```

```
#define AFX_WINANN_H__0E253D02_EFB8_11D3_8089_0090276F4183__INCLUDED_

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

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

#include "resource.h"// main symbols

////////////////////////////////////////////////////////////////////////////
// CWinANNApp:
// See WinANN.cpp for the implementation of this class
//

class CWinANNApp : public CWinApp
{
public:
    CWinANNApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWinANNApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation

    //{{AFX_MSG(CWinANNApp)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_WINANN_H__0E253D02_EFB8_11D3_8089_0090276F4183__INCLUDED_)
```

D.10 WinANNDlg.cpp

```
// WinANNDlg.cpp : implementation file
//

#include "stdafx.h"
#include <math.h>
#include "WinANN.h"
#include "WinANNDlg.h"
#include "back_prop.h"
#include "ConfigDlg.h"


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


////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
```

```
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWinANNDlg dialog

CWinANNDlg::CWinANNDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CWinANNDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CWinANNDlg)
    m_ErrorLimit = 0.0f;
    m_IterPerBeta = 0;
    m_IL = 0;
    m_ML = 0;
    m_OL = 0;
    m_StartBeta = 0.0f;
    m_StopBeta = 0.0f;
    m_CurrBeta = 0.0f;
    m_CurrError = 0.0f;
    m_MaxIterations = 0;
    m_MaxError = 0.0f;
    m_TrainMode = 0;
    m_CurrIteration = 0;
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CWinANNDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CWinANNDlg)
```

```
    DDX_Control(pDX, IDC_TRAINMORE, m_TrainMoreButton);
    DDX_Control(pDX, IDC_TEST, m_TestButton);
    DDX_Control(pDX, IDCANCEL, m_CancelButton);
    DDX_Control(pDX, IDOK, m_OkButton);
    DDX_Control(pDX, IDC_PAUSE, m_PauseButton);
    DDX_Control(pDX, IDC_GO, m_GoButton);
    DDX_Text(pDX, IDC_ERRORLIMIT, m_ErrorLimit);
    DDV_MinMaxFloat(pDX, m_ErrorLimit, 1.e-003f, 100.f);
    DDX_Text(pDX, IDC_ITPERBETA, m_IterPerBeta);
    DDV_MinMaxInt(pDX, m_IterPerBeta, 1, 10000);
    DDX_Text(pDX, IDC_IL, m_IL);
    DDV_MinMaxInt(pDX, m_IL, 1, 10000);
    DDX_Text(pDX, IDC_ML, m_ML);
    DDV_MinMaxInt(pDX, m_ML, 2, 10000);
    DDX_Text(pDX, IDC_OL, m_OL);
    DDV_MinMaxInt(pDX, m_OL, 1, 10000);
    DDX_Text(pDX, IDC_STARTBETA, m_StartBeta);
    DDV_MinMaxFloat(pDX, m_StartBeta, 1.e-002f, 1.f);
    DDX_Text(pDX, IDC_STOPBETA, m_StopBeta);
    DDV_MinMaxFloat(pDX, m_StopBeta, 1.e-002f, 1.f);
    DDX_Text(pDX, IDC_CURRBETA, m_CurrBeta);
    DDV_MinMaxFloat(pDX, m_CurrBeta, 0.f, 100.f);
    DDX_Text(pDX, IDC_CURRERROR, m_CurrError);
    DDV_MinMaxFloat(pDX, m_CurrError, 1.e-008f, 1.e+006f);
    DDX_Text(pDX, IDC_MAXITER, m_MaxIterations);
    DDV_MinMaxInt(pDX, m_MaxIterations, 1, 10000);
    DDX_Text(pDX, IDC_MAXERROR, m_MaxError);
    DDV_MinMaxFloat(pDX, m_MaxError, 1.e-003f, 1000.f);
    DDX_Text(pDX, IDC_TRAINING, m_TrainMode);
    DDV_MinMaxInt(pDX, m_TrainMode, 0, 2);
    DDX_Text(pDX, IDC_CURRITER, m_CurrIteration);
    DDV_MinMaxInt(pDX, m_CurrIteration, 0, 10000);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CWinANNDlg, CDialog)
    //{{AFX_MSG_MAP(CWinANNDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_GO, OnGo)
    ON_BN_CLICKED(IDC_PAUSE, OnPause)
    ON_WM_DESTROY()
    ON_BN_CLICKED(IDC_TEST, OnTest)
    ON_BN_CLICKED(IDC_TRAINMORE, OnTrainmore)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWinANNDlg message handlers

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

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }
```

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

    CWnd * pCanvas = GetDlgItem(IDC_GRAPH);
    pCanvas->GetWindowRect(&m_Canvas);
    ScreenToClient(&m_Canvas);
    m_Canvas.DeflateRect(2,2,1,1);

    m_BPThread = NULL;
    m_PausedThread = TRUE;
    m_GoButton.EnableWindow(TRUE);
    m_PauseButton.EnableWindow(FALSE);
    m_OkButton.EnableWindow(FALSE);
    m_CancelButton.EnableWindow(TRUE);
    m_TestButton.EnableWindow(FALSE);
    m_TrainMoreButton.EnableWindow(FALSE);

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

void CWinANNDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}


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

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

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

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

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        //CDialog::OnPaint();
        CPaintDC pDC(this); // device context for painting


        if (m_BPThread)
        {
            pDC.SelectStockObject(NULL_PEN);
            pDC.Rectangle(m_Canvas);
            pDC.SelectStockObject(BLACK_BRUSH);
            pDC.SelectStockObject(BLACK_PEN);
```

```
            if (m_BPThread->m_NewData)
            {
                m_CurrBeta = m_BPThread->m_CurrBeta;
                m_CurrError = m_BPThread->m_CurrError;
                m_CurrIteration = m_BPThread->m_CurrIteration;
                UpdateData(FALSE);//write to window

                int nPoints = m_BPThread->NumPoints();
                if (nPoints>1)
                {
                    for (int i = 0; i < nPoints-1; i++)
                    {
                        //convert to window coordinates...
                        pDC.MoveTo(m_BPThread->GetPoint(i));
                        pDC.LineTo(m_BPThread->GetPoint(i+1));
                    }
                }
            }

            if (m_BPThread->m_Done)
            {
                m_BPThread->m_Done = FALSE;
                if (m_BPThread->m_TrainMode==0)
                    AfxMessageBox("Done Testing");
                m_GoButton.EnableWindow(FALSE);//Don't want to re-run
                m_PauseButton.EnableWindow(FALSE);
                m_OkButton.EnableWindow(TRUE);
                m_CancelButton.EnableWindow(FALSE);
                m_TestButton.EnableWindow(TRUE);
                m_TrainMoreButton.EnableWindow(TRUE);
            }
        }//end if m_BPThread
        else
        {
            //clear canvas white
            pDC.SelectStockObject(NULL_PEN);
            pDC.Rectangle(m_Canvas);
        }
    }//end else not minimized
}//end funciton Paint()

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


UINT MyThreadProc(LPVOID pParam)
{

    back_prop * pObject = (back_prop*)pParam;

    if (pObject == NULL || !pObject->IsKindOf(RUNTIME_CLASS(back_prop)))
    {
        AfxMessageBox("Thread failed");
        return 1;   // if pObject is not valid
    }

    // do something with 'pObject'
    AfxMessageBox("here we go");
    pObject->go();
    AfxMessageBox("done thread");
    return 0;   // thread completed successfully

}

void CWinANNDlg::OnGo()
{
    //back_prop * pNewBackProp = new back_prop;
```

```
    if (!m_BPThread)
    {

        //user interface thread create suspended so we can init some of the values for it
        m_BPThread = (back_prop *)AfxBeginThread(RUNTIME_CLASS(back_prop), THREAD_PRIORITY_NORMAL, 0,
CREATE_SUSPENDED);

        m_BPThread->m_ErrorLimit = m_ErrorLimit;
        m_BPThread->m_IterPerBeta = m_IterPerBeta;
        m_BPThread->m_IL = m_IL;
        m_BPThread->m_ML = m_ML;
        m_BPThread->m_OL = m_OL;
        m_BPThread->m_StartBeta = m_StartBeta;
        m_BPThread->m_StopBeta = m_StopBeta;
        m_BPThread->m_CurrBeta = m_CurrBeta;
        m_BPThread->m_CurrError = m_CurrError;
        m_BPThread->m_MaxIterations = m_MaxIterations;
        m_BPThread->m_MaxError = m_MaxError;
        m_BPThread->m_TrainMode = m_TrainMode;
        m_BPThread->m_CurrIteration = m_CurrIteration;
        m_BPThread->m_DlgWnd = GetDlgItem(IDC_GRAPH)->GetParent();
        m_BPThread->m_Canvas = m_Canvas;
        m_BPThread->m_OldMaxIterations = 0;
    }

    //make sure it was created successfully
    if (!m_BPThread)
    {
        AfxMessageBox("Thread could not be created");
    }
    else
    {
        m_PausedThread = FALSE;
        m_BPThread->ResumeThread();
        m_GoButton.EnableWindow(FALSE);
        m_PauseButton.EnableWindow(TRUE);
    }//end else
}//end GO()


void CWinANNDlg::OnPause()
{
    // carry out pausing actions (mainly, suspend the thread
    m_PausedThread = TRUE;
    m_BPThread->SuspendThread();
    m_GoButton.EnableWindow(TRUE);
    m_PauseButton.EnableWindow(FALSE);
}

void CWinANNDlg::OnDestroy()
{
    //needed to override the destroy function to get rid of paused ui thread
    CDialog::OnDestroy();
    if (m_PausedThread && m_BPThread)
        m_BPThread->ResumeThread();
}

void CWinANNDlg::OnTest()
{
    //can just rerun the network changing train to 0;
    m_TrainMode = 0;
    m_BPThread->m_Done = FALSE;
    m_BPThread->m_TrainMode = 0;
    m_BPThread->go();
}

void CWinANNDlg::OnTrainmore()
{
    //get the configuration information
    CConfigDlg Cdlg;
    Cdlg.m_ErrorLimit = 0.1;
```

```
    Cdlg.m_IterPerBeta = 500;
    Cdlg.m_StartBeta = 0.75;
    Cdlg.m_StopBeta = 0.01;
    Cdlg.m_IL = m_IL;
    Cdlg.m_ML = m_ML;
    Cdlg.m_OL = m_OL;
    Cdlg.m_TrainMode = 2;
    Cdlg.m_TrainMore = TRUE;

    if (Cdlg.DoModal() == IDOK)
    {
        m_ErrorLimit = Cdlg.m_ErrorLimit;
        m_IterPerBeta = Cdlg.m_IterPerBeta;
        m_StartBeta = Cdlg.m_StartBeta;
        m_StopBeta = Cdlg.m_StopBeta;
        m_TrainMode = 2;
        m_BPThread->m_ErrorLimit = m_ErrorLimit;
        m_BPThread->m_IterPerBeta = m_IterPerBeta;
        m_BPThread->m_StartBeta = m_StartBeta;
        m_BPThread->m_StopBeta = m_StopBeta;
        m_BPThread->m_TrainMode = m_TrainMode;

        //calculate the x size of the canvas graph
        double TempMI=0.0f;
        TempMI = log((double)m_StopBeta/(double)m_StartBeta);
        TempMI = TempMI/log(0.75);
        TempMI += 1.5f;
        TempMI *= 1.1;
        TempMI = floor(TempMI);
        TempMI *= m_IterPerBeta;
        TempMI += m_MaxIterations;

        m_BPThread->m_OldMaxIterations = m_MaxIterations;

        //change all of the points to new x values:
        for (int i=0;i<m_BPThread->NumPoints();i++)
        {
            m_BPThread->m_Data[i].x = ((m_BPThread->GetPoint(i).x-m_Canvas.left) * m_MaxIterations)/
TempMI + m_Canvas.left;
        }

        m_MaxIterations = TempMI;
        m_CurrBeta = m_StartBeta;
        m_CurrError = 0.001;
        m_MaxError = 50.0;
        m_CurrIteration = 0;
        UpdateData(FALSE);//write to window

        // should go automatically because that is what the user wants...
        m_BPThread->m_MaxIterations = m_MaxIterations;
        m_BPThread->m_Done = FALSE;
        m_GoButton.EnableWindow(TRUE);
        m_PauseButton.EnableWindow(FALSE);
        m_OkButton.EnableWindow(FALSE);
        m_CancelButton.EnableWindow(TRUE);
        m_TestButton.EnableWindow(FALSE);
        m_TrainMoreButton.EnableWindow(FALSE);
        m_PausedThread = TRUE;
        m_BPThread->SuspendThread();
        m_BPThread->go();
        //m_BPThread->ResumeThread();
        AfxMessageBox("Thread done");
    }
}
```

D.11 WinAnnDlg.h

```
// WinANNDlg.h : header file
```

```
//

#if !defined(AFX_WINANNDLG_H__0E253D04_EFB8_11D3_8089_0090276F4183__INCLUDED_)
#define AFX_WINANNDLG_H__0E253D04_EFB8_11D3_8089_0090276F4183__INCLUDED_

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

#include "back_prop.h"


/////////////////////////////////////////////////////////////////////////////
// CWinANNDlg dialog

class CWinANNDlg : public CDialog
{
// Construction
public:
    back_prop * m_BPThread;
    CRect m_Canvas;
    CWinANNDlg(CWnd* pParent = NULL);// standard constructor
    BOOL m_PausedThread;

// Dialog Data
    //{{AFX_DATA(CWinANNDlg)
    enum { IDD = IDD_WINANN_DIALOG };
    CButtonm_TrainMoreButton;
    CButtonm_TestButton;
    CButtonm_CancelButton;
    CButtonm_OkButton;
    CButtonm_PauseButton;
    CButtonm_GoButton;
    floatm_ErrorLimit;
    int     m_IterPerBeta;
    int     m_IL;
    int     m_ML;
    int     m_OL;
    floatm_StartBeta;
    floatm_StopBeta;
    floatm_CurrBeta;
    floatm_CurrError;
    int     m_MaxIterations;
    floatm_MaxError;
    int     m_TrainMode;
    int     m_CurrIteration;
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWinANNDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);// DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    HICON m_hIcon;

    // Generated message map functions
    //{{AFX_MSG(CWinANNDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnGo();
    afx_msg void OnPause();
    afx_msg void OnDestroy();
    afx_msg void OnTest();
    afx_msg void OnTrainmore();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

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

#endif // !defined(AFX_WINANNDLG_H__0E253D04_EFB8_11D3_8089_0090276F4183__INCLUDED_)
```

APPENDIX E

BOWANN SOURCE FILES

E.1 AddPointSourceDlg.cpp

```
// AddPointSourceDlg.cpp : implementation file
//

#include "stdafx.h"
#include "isocurve.h"
#include "AddPointSourceDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CAddPointSourceDlg dialog


CAddPointSourceDlg::CAddPointSourceDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CAddPointSourceDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CAddPointSourceDlg)
    m_X = 0;
    m_Y = 0;
    m_DwellTime = 0;
    //}}AFX_DATA_INIT
}


void CAddPointSourceDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAddPointSourceDlg)
    DDX_Text(pDX, IDC_X, m_X);
    DDV_MinMaxInt(pDX, m_X, 0, 1000);
    DDX_Text(pDX, IDC_Y, m_Y);
    DDV_MinMaxInt(pDX, m_Y, 0, 1000);
    DDX_Text(pDX, IDC_DWELLTIME, m_DwellTime);
    DDV_MinMaxInt(pDX, m_DwellTime, 1, 1000);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CAddPointSourceDlg, CDialog)
    //{{AFX_MSG_MAP(CAddPointSourceDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAddPointSourceDlg message handlers
```

E.2 AddPointSourceDlg.h

```
#if !defined(AFX_ADDPOINTSOURCEDLG_H__D886E81D_AE77_11D3_8082_0090276F4183__INCLUDED_)
```

```
#define AFX_ADDPOINTSOURCEDLG_H__D886E81D_AE77_11D3_8082_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// AddPointSourceDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CAddPointSourceDlg dialog

class CAddPointSourceDlg : public CDialog
{
// Construction
public:
    CAddPointSourceDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CAddPointSourceDlg)
    enum { IDD = IDD_ADDPOINTSOURCE };
    int     m_X;
    int     m_Y;
    int     m_DwellTime;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAddPointSourceDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CAddPointSourceDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_ADDPOINTSOURCEDLG_H__D886E81D_AE77_11D3_8082_0090276F4183__INCLUDED_)
```

E.3 AddTypeDlg.cpp

```
// AddTypeDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "AddTypeDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CAddTypeDlg dialog


CAddTypeDlg::CAddTypeDlg(CWnd* pParent /*=NULL*/)
```

```
    : CDialog(CAddTypeDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CAddTypeDlg)
    m_AddType = -1;
    //}}AFX_DATA_INIT
}


void CAddTypeDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAddTypeDlg)
    DDX_Radio(pDX, IDC_POINT, m_AddType);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CAddTypeDlg, CDialog)
    //{{AFX_MSG_MAP(CAddTypeDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAddTypeDlg message handlers
```

E.4 AddTypeDlg.h

```
#if !defined(AFX_ADDTYPEDLG_H__ADFB47E0_5A31_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_ADDTYPEDLG_H__ADFB47E0_5A31_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// AddTypeDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CAddTypeDlg dialog

class CAddTypeDlg : public CDialog
{
// Construction
public:
    CAddTypeDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CAddTypeDlg)
    enum { IDD = IDD_ADDTYPE };
    int     m_AddType;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAddTypeDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CAddTypeDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

```
};

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

#endif // !defined(AFX_ADDTYPEDLG_H__ADFB47E0_5A31_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.5 Backprop.cpp

```
// backprop.cppV. Rao, H. Rao
#include "layer.cpp"

#define TRAINING_FILE"training.dat"
#define WEIGHTS_FILE "weights.dat"
#define OUTPUT_FILE"output.dat"
#define TEST_FILE"test.dat"

void main()
{
float error_tolerance=0.1;
float total_error=0.0;
float avg_error_per_cycle=0.0;
float error_last_cycle=0.0;
float avgerr_per_pattern=0.0; // for the latest cycle
float error_last_pattern=0.0;
float learning_parameter=0.02;
unsigned temp, startup;
long int vectors_in_buffer;
long int max_cycles;
long int patterns_per_cycle=0;

long int total_cycles, total_patterns;
int i;


// create a network object
network backp;


FILE * training_file_ptr, * weights_file_ptr, * output_file_ptr, * p_MyOutputFile;
FILE * test_file_ptr, * data_file_ptr;
int m_MoreTrain=0;

// open output file for writing
if ((output_file_ptr=fopen(OUTPUT_FILE,"w"))==NULL)
        {
        cout << "problem opening output file\n";
        exit(1);
        }

// open my output file for writing
if ((p_MyOutputFile=fopen("MyOutput.dat","w"))==NULL)
        {
        cout << "problem opening my output file\n";
        exit(1);
        }


// enter the training mode : 1=training on     0=training off
cout << "-------------------------------------------------\n";
cout << " C++ Neural Networks and Fuzzy Logic \n";
cout << " Backpropagation simulator \n";
cout << "version 1 \n";
cout << "-------------------------------------------------\n";
cout << "Please enter 2 for MORE TRAINING, 1 for TRAINING on, 0 for off: \n\n";
cout << "Use training to change weights according to your\n";
cout << "expected outputs. Your training.dat file should contain\n";
cout << "a set of inputs and expected outputs. The number of\n";
```

```
cout << "inputs determines the size of the first (input) layer\n";
cout << "while the number of outputs determines the size of the\n";
cout << "last (output) layer :\n\n";

cin >> temp;
if (temp == 2)
{
    // set the global flag for loading weights instead of random
    m_MoreTrain = 1;
    temp = 1;// so that it treats all else as training
}
else
    m_MoreTrain = 0;
backp.set_training(temp);

if (backp.get_training_value() == 1)
    {
    cout << "--> Training mode is *ON*. weights will be saved\n";
    cout << "in the file weights.dat at the end of the\n";
    cout << "current set of input (training) data\n";
    }
else
    {
    cout << "--> Training mode is *OFF*. weights will be loaded\n";
    cout << "from the file weights.dat and the current\n";
    cout << "(test) data set will be used. For the test\n";
    cout << "data set, the test.dat file should contain\n";
    cout << "only inputs, and no expected outputs.\n";
    }

if (backp.get_training_value()==1)
    {
    // ----------------------------------------
    //  Read in values for the error_tolerance,
    //  and the learning_parameter
    // ----------------------------------------
    cout << " Please enter in the error_tolerance\n";
    cout << " --- between 0.001 to 100.0, try 0.1 to start --\n";
    cout << "\n";
    cout << "and the learning_parameter, beta\n";
    cout << " --- between 0.01 to 1.0, try 0.5 to start -- \n\n";
    cout << " separate entries by a space\n";
    cout << " example: 0.1 0.5 sets defaults mentioned :\n\n";

    cin >> error_tolerance >> learning_parameter;
    //-----------------------------------------
    // open training file for reading
    //-----------------------------------------
    if ((training_file_ptr=fopen(TRAINING_FILE,"r"))==NULL)
        {
        cout << "problem opening training file\n";
        exit(1);
        }
    data_file_ptr=training_file_ptr; // training on

    // Read in the maximum number of cycles
    // each pass through the input data file is a cycle
    cout << "Please enter the maximum cycles for the simulation\n";
    cout << "A cycle is one pass through the data set.\n";
    cout << "Try a value of 10 to start with\n";

    cin >> max_cycles;

    }
else
    {
    if ((test_file_ptr=fopen(TEST_FILE,"r"))==NULL)
        {
        cout << "problem opening test file\n";
        exit(1);
        }
```

```
    data_file_ptr=test_file_ptr; // training off
    }



// the main loop
//
// training: continue looping until the total error is less than
//      the tolerance specified, or the maximum number of
//      cycles is exceeded; use both the forward signal propagation
//      and the backward error propagation phases. If the error
//      tolerance criteria is satisfied, save the weights in a file.
// no training: just proceed through the input data set once in the
//      forward signal propagation phase only. Read the starting
//      weights from a file.
// in both cases report the outputs on the screen


// intialize counters
total_cycles=0; // a cycle is once through all the input data
total_patterns=0; // a pattern is one entry in the input data



// get layer information
backp.get_layer_info();

// set up the network connections
backp.set_up_network();

// initialize the weights
if (backp.get_training_value()==1)
    {
    if (!m_MoreTrain)// them just make random weights
    {
        // randomize weights for all layers; there is no
        // weight matrix associated with the input layer
        // weight file will be written after processing
        // so open for writing
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))
                ==NULL)
            {
            cout << "problem opening weights file\n";
            exit(1);
            }
        backp.randomize_weights();
    }
    else//otherwise load in weights from the past
    {
        cout << "Loading weights from file\n";
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))==NULL)
        {
            cout << "problem opening weights file\n";
            exit(1);
        }
        backp.read_weights(weights_file_ptr);
        fclose(weights_file_ptr);
        cout << "Weights loaded, no problems\n";
        // now reopen weights file to be rewritten
        if ((weights_file_ptr=fopen(WEIGHTS_FILE,"w"))==NULL)
        {
            cout << "problem opening weights file\n";
            exit(1);
        }
    }
    }
else
    {
    // read in the weight matrix defined by a
    // prior run of the backpropagation simulator
    // with training on
```

```
    if ((weights_file_ptr=fopen(WEIGHTS_FILE,"r"))
            ==NULL)
        {
        cout << "problem opening weights file\n";
        exit(1);
        }
    backp.read_weights(weights_file_ptr);
    }



// main loop
// if training is on, keep going through the input data
//      until the error is acceptable or the maximum number of cycles
//      is exceeded.
// if training is off, go through the input data once. report outputs
// with inputs to file output.dat

startup=1;
vectors_in_buffer = MAX_VECTORS; // startup condition
total_error = 0;


while (     ((backp.get_training_value()==1)
            && (avgerr_per_pattern
                    > error_tolerance)
            && (total_cycles < max_cycles)
            && (vectors_in_buffer !=0))
            || ((backp.get_training_value()==0)
            && (total_cycles < 1))
            || ((backp.get_training_value()==1)
            && (startup==1))
            )
{
startup=0;
error_last_cycle=0; // reset for each cycle
patterns_per_cycle=0;
// process all the vectors in the datafile
// going through one buffer at a time
// pattern by pattern


while ((vectors_in_buffer==MAX_VECTORS))
    {

    vectors_in_buffer=
        backp.fill_IObuffer(data_file_ptr); // fill buffer
        if (vectors_in_buffer < 0)
            {
            cout << "error in reading in vectors, aborting\n";
            cout << "check that there are on extra linefeeds\n";
            cout << "in your data file, and that the number\n";
            cout << "of layers and size of layers match the\n";
            cout << "the parameters provided.\n";
            exit(1);
            }

        // process vectors
        for (i=0; i<vectors_in_buffer; i++)
            {
            // get next pattern
            backp.set_up_pattern(i);

            total_patterns++;
            patterns_per_cycle++;
            // forward propagate

            backp.forward_prop();

            if (backp.get_training_value()==0)
```

```
                backp.write_outputs(output_file_ptr,p_MyOutputFile);

            // back_propagate, if appropriate
            if (backp.get_training_value()==1)
                {

                backp.backward_prop(error_last_pattern);
                error_last_cycle +=
                    error_last_pattern*error_last_pattern;
                backp.update_weights(learning_parameter);
                // backp.list_weights(); // can
                // see change in weights by
                // using list_weights before and
                // after back_propagation
                }

            }

    error_last_pattern = 0;
    }

avgerr_per_pattern=((float)sqrt((double)error_last_cycle/patterns_per_cycle));
total_error += error_last_cycle;
total_cycles++;

// most character displays are 25 lines
// user will see a corner display of the cycle count
// as it changes

cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
cout << total_cycles << "\t" << avgerr_per_pattern << "\n";

fseek(data_file_ptr, 0L, SEEK_SET); // reset the file pointer
                // to the beginning of
                // the file
vectors_in_buffer = MAX_VECTORS; // reset

} // end main loop

cout << "\n\n\n\n\n\n\n\n\n\n\n";
cout << "--------------------------------------------------\n";
cout << " done:results in file output.dat\n";
cout << "training: last vector only\n";
cout << "not training: full cycle\n\n";
if (backp.get_training_value()==1)
    {
    backp.write_weights(weights_file_ptr);
    backp.write_outputs(output_file_ptr,p_MyOutputFile);
    avg_error_per_cycle= (float)sqrt((double)total_error/total_cycles);
    error_last_cycle=(float)sqrt((double)error_last_cycle);

cout << "weights saved in file weights.dat\n";
cout << "\n";
cout << "---->average error per cycle = " << avg_error_per_cycle << " <---\n";
cout << "---->error last cycle = " << error_last_cycle << " <---\n";
cout << "->error last cycle per pattern= " << avgerr_per_pattern << " <---\n";

    }

cout << "------------>total cycles = " << total_cycles << " <---\n";
cout << "------------>total patterns = " << total_patterns << " <---\n";
cout << "--------------------------------------------------\n";
// close all files
fclose(data_file_ptr);
fclose(weights_file_ptr);
fclose(output_file_ptr);
fclose(p_MyOutputFile);
}
```

E.6 ChildFrm.cpp

```
// ChildFrm.cpp : implementation of the CChildFrame class
//

#include "stdafx.h"
#include "Isocurve.h"

#include "ChildFrm.h"

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

/////////////////////////////////////////////////////////////////////////////
// CChildFrame

IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
    //{{AFX_MSG_MAP(CChildFrame)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CChildFrame construction/destruction

CChildFrame::CChildFrame()
{
    // TODO: add member initialization code here

}

CChildFrame::~CChildFrame()
{
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    if( !CMDIChildWnd::PreCreateWindow(cs) )
        return FALSE;

    return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CChildFrame diagnostics

#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
    CMDIChildWnd::AssertValid();
}

void CChildFrame::Dump(CDumpContext& dc) const
{
    CMDIChildWnd::Dump(dc);
}

#endif //_DEBUG

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

```
// CChildFrame message handlers
```

E.7 ChildFrm.h

```
// ChildFrm.h : interface of the CChildFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CHILDFRM_H__E8F0A3AA_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_CHILDFRM_H__E8F0A3AA_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_

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


class CChildFrame : public CMDIChildWnd
{
    DECLARE_DYNCREATE(CChildFrame)
public:
    CChildFrame();

// Attributes
public:

// Operations
public:

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

// Implementation
public:
    virtual ~CChildFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CChildFrame)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_CHILDFRM_H__E8F0A3AA_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.8 CntrlItem.cpp

```
// CntrItem.cpp : implementation of the CIsocurveCntrItem class
//
```

```
#include "stdafx.h"
#include "Isocurve.h"

#include "IsocurveDoc.h"
#include "IsocurveView.h"
#include "CntrItem.h"

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

/////////////////////////////////////////////////////////////////////////////
// CIsocurveCntrItem implementation

IMPLEMENT_SERIAL(CIsocurveCntrItem, COleClientItem, 0)

CIsocurveCntrItem::CIsocurveCntrItem(CIsocurveDoc* pContainer)
    : COleClientItem(pContainer)
{
    // TODO: add one-time construction code here

}

CIsocurveCntrItem::~CIsocurveCntrItem()
{
    // TODO: add cleanup code here
}

void CIsocurveCntrItem::OnChange(OLE_NOTIFICATION nCode, DWORD dwParam)
{
    ASSERT_VALID(this);

    COleClientItem::OnChange(nCode, dwParam);

    // When an item is being edited (either in-place or fully open)
    //  it sends OnChange notifications for changes in the state of the
    //  item or visual appearance of its content.

    // TODO: invalidate the item by calling UpdateAllViews
    //  (with hints appropriate to your application)

    GetDocument()->UpdateAllViews(NULL);
        // for now just update ALL views/no hints
}

BOOL CIsocurveCntrItem::OnChangeItemPosition(const CRect& rectPos)
{
    ASSERT_VALID(this);

    // During in-place activation CIsocurveCntrItem::OnChangeItemPosition
    //  is called by the server to change the position of the in-place
    //  window.  Usually, this is a result of the data in the server
    //  document changing such that the extent has changed or as a result
    //  of in-place resizing.
    //
    // The default here is to call the base class, which will call
    //  COleClientItem::SetItemRects to move the item
    //  to the new position.

    if (!COleClientItem::OnChangeItemPosition(rectPos))
        return FALSE;

    // TODO: update any cache you may have of the item's rectangle/extent

    return TRUE;
}

void CIsocurveCntrItem::OnGetItemPosition(CRect& rPosition)
{
    ASSERT_VALID(this);
```

```
    // During in-place activation, CIsocurveCntrItem::OnGetItemPosition
    //  will be called to determine the location of this item.  The default
    //  implementation created from AppWizard simply returns a hard-coded
    //  rectangle.  Usually, this rectangle would reflect the current
    //  position of the item relative to the view used for activation.
    //  You can obtain the view by calling CIsocurveCntrItem::GetActiveView.

    // TODO: return correct rectangle (in pixels) in rPosition

    rPosition.SetRect(10, 10, 210, 210);
}

void CIsocurveCntrItem::OnActivate()
{
    // Allow only one inplace activate item per frame
    CIsocurveView* pView = GetActiveView();
    ASSERT_VALID(pView);
    COleClientItem* pItem = GetDocument()->GetInPlaceActiveItem(pView);
    if (pItem != NULL && pItem != this)
        pItem->Close();

    COleClientItem::OnActivate();
}

void CIsocurveCntrItem::OnDeactivateUI(BOOL bUndoable)
{
    COleClientItem::OnDeactivateUI(bUndoable);

    // Hide the object if it is not an outside-in object
    DWORD dwMisc = 0;
    m_lpObject->GetMiscStatus(GetDrawAspect(), &dwMisc);
    if (dwMisc & OLEMISC_INSIDEOUT)
        DoVerb(OLEIVERB_HIDE, NULL);
}

void CIsocurveCntrItem::Serialize(CArchive& ar)
{
    ASSERT_VALID(this);

    // Call base class first to read in COleClientItem data.
    // Since this sets up the m_pDocument pointer returned from
    //  CIsocurveCntrItem::GetDocument, it is a good idea to call
    //  the base class Serialize first.
    COleClientItem::Serialize(ar);

    // now store/retrieve data specific to CIsocurveCntrItem
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}

/////////////////////////////////////////////////////////////////////////////
// CIsocurveCntrItem diagnostics

#ifdef _DEBUG
void CIsocurveCntrItem::AssertValid() const
{
    COleClientItem::AssertValid();
}

void CIsocurveCntrItem::Dump(CDumpContext& dc) const
{
    COleClientItem::Dump(dc);
}
#endif
```

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

E.9 CntrlItem.h

```
// CntrItem.h : interface of the CIsocurveCntrItem class
//

#if !defined(AFX_CNTRITEM_H__E8F0A3B0_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_CNTRITEM_H__E8F0A3B0_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_

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

class CIsocurveDoc;
class CIsocurveView;

class CIsocurveCntrItem : public COleClientItem
{
    DECLARE_SERIAL(CIsocurveCntrItem)

// Constructors
public:
    CIsocurveCntrItem(CIsocurveDoc* pContainer = NULL);
        // Note: pContainer is allowed to be NULL to enable IMPLEMENT_SERIALIZE.
        //  IMPLEMENT_SERIALIZE requires the class have a constructor with
        //  zero arguments.  Normally, OLE items are constructed with a
        //  non-NULL document pointer.

// Attributes
public:
    CIsocurveDoc* GetDocument()
        { return (CIsocurveDoc*)COleClientItem::GetDocument(); }
    CIsocurveView* GetActiveView()
        { return (CIsocurveView*)COleClientItem::GetActiveView(); }

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CIsocurveCntrItem)
    public:
    virtual void OnChange(OLE_NOTIFICATION wNotification, DWORD dwParam);
    virtual void OnActivate();
    protected:
    virtual void OnGetItemPosition(CRect& rPosition);
    virtual void OnDeactivateUI(BOOL bUndoable);
    virtual BOOL OnChangeItemPosition(const CRect& rectPos);
    //}}AFX_VIRTUAL

// Implementation
public:
    ~CIsocurveCntrItem();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
    virtual void Serialize(CArchive& ar);
};

////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_CNTRITEM_H__E8F0A3B0_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.10 CoolingDoc.cpp

```
// CoolingDoc.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "CoolingDoc.h"

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

/////////////////////////////////////////////////////////////////////////////
// CCoolingDoc

IMPLEMENT_DYNCREATE(CCoolingDoc, COleDocument)

CCoolingDoc::CCoolingDoc()
{
}

BOOL CCoolingDoc::OnNewDocument()
{
    if (!COleDocument::OnNewDocument())
        return FALSE;
    return TRUE;
}

CCoolingDoc::~CCoolingDoc()
{
}


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

/////////////////////////////////////////////////////////////////////////////
// CCoolingDoc diagnostics

#ifdef _DEBUG
void CCoolingDoc::AssertValid() const
{
    COleDocument::AssertValid();
}

void CCoolingDoc::Dump(CDumpContext& dc) const
{
    COleDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CCoolingDoc serialization

void CCoolingDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CCoolingDoc commands
```

E.11 CoolingDoc.h

```
#if !defined(AFX_COOLINGDOC_H__D5E2CBA1_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_COOLINGDOC_H__D5E2CBA1_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// CoolingDoc.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CCoolingDoc document

class CCoolingDoc : public COleDocument
{
protected:
    CCoolingDoc();           // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CCoolingDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CCoolingDoc)
    public:
    virtual void Serialize(CArchive& ar);   // overridden for document i/o
    protected:
    virtual BOOL OnNewDocument();
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CCoolingDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    // Generated message map functions
protected:
    //{{AFX_MSG(CCoolingDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_COOLINGDOC_H__D5E2CBA1_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.12 CoolingView.cpp

```
// CoolingView.cpp : implementation file
```

```
//

#include "stdafx.h"
#include "Isocurve.h"
#include "CoolingView.h"

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

/////////////////////////////////////////////////////////////////////////////
// CCoolingView

IMPLEMENT_DYNCREATE(CCoolingView, CView)

CCoolingView::CCoolingView()
{
}

CCoolingView::~CCoolingView()
{
}


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

/////////////////////////////////////////////////////////////////////////////
// CCoolingView drawing

void CCoolingView::OnDraw(CDC* pDC)
{
    CDocument* pDoc = GetDocument();
    // TODO: add draw code here
}

/////////////////////////////////////////////////////////////////////////////
// CCoolingView diagnostics

#ifdef _DEBUG
void CCoolingView::AssertValid() const
{
    CView::AssertValid();
}

void CCoolingView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CCoolingView message handlers
```

E.13 CoolingView.h

```
#if !defined(AFX_COOLINGVIEW_H__D5E2CBA2_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_COOLINGVIEW_H__D5E2CBA2_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// CoolingView.h : header file
```

```
//

/////////////////////////////////////////////////////////////////////////////
// CCoolingView view

class CCoolingView : public CView
{
protected:
    CCoolingView();           // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CCoolingView)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CCoolingView)
    protected:
    virtual void OnDraw(CDC* pDC);      // overridden to draw this view
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CCoolingView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    // Generated message map functions
protected:
    //{{AFX_MSG(CCoolingView)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_COOLINGVIEW_H__D5E2CBA2_44C6_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.14 Cost.c

```
#include <stdio.h>
#include <math.h>
#include "globals.h"


int CheckHyperDoseSleave(PS *ThePS)
{
int i;
int j;
float Dose=0.0f;
PS *TempPS=NULL;
PS *TempStaticPS=NULL;
float Min=9999999.f;

/*Go through the whole "Area" and calculate the dose*/
    for (i=0;i<xSize;i++)
    {
        for (j=0;j<ySize;j++)
```

```
        {
            // Run though Dynamic list of Sources
                Dose = 0.0f;
                TempPS = ThePS;
                while (TempPS)
                {
                    if ((TempPS -> x == i) && (TempPS -> y == j))
                        Dose += TempPS -> DwellTime*sqr(RATIO);
                    else
                        Dose += ((TempPS -> DwellTime)*sqr(RATIO))/sqr(distance(i,j,TempPS->x,TempPS-
>y));
                    TempPS = TempPS -> NextPS;
                }// end while
            // Run through Static list of Sources
                TempStaticPS = FirstStaticPS;
                while (TempStaticPS)
                {
                    if ((TempStaticPS -> x == i) && (TempStaticPS -> y == j))
                        Dose += TempStaticPS -> DwellTime * sqr(RATIO);
                    else
                        Dose += ((TempStaticPS -> DwellTime)*sqr(RATIO))/sqr(distance(i,j,TempStaticPS-
>x,TempStaticPS->y));

                    TempStaticPS = TempStaticPS -> NextPS;
                }// end while
            // Store the Dose to the "Area" array
                (Area + ySize*i + j) -> Dose = Dose;

            // might as well store the minimum to the periphery at the same time
                if (((Area + ySize*i + j) -> Type == PERIPHERY)&&(Dose<Min))
                    Min = Dose;
        }
    }


//check that the hyperdose sleave is small enough for Dynamic Sources
    TempPS = ThePS;
    while (TempPS)
    {
        j = TempPS -> y;
        i = TempPS -> x;
        /* west */
        if (j>5*RATIO)
        {
            Dose = (((Area + ySize*i + (j-5*RATIO))->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else //j=0
        {
            Dose = (((Area + ySize*i)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* north */
        if (i>5*RATIO)
        {
            Dose = (((Area + ySize*(i-5*RATIO) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else //i=0
        {
            Dose = (((Area + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* east */
        if (j+5*RATIO<ySize)
        {
            Dose = (((Area + ySize*i + (j+5*RATIO))->Dose)/Min);
            if (Dose>2.f)
```

```
            return(1);
        }
        else //j=ySize-1
        {
            Dose = (((Area + ySize*i + (ySize-1))->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* south */
        if (i+5*RATIO<xSize)
        {
            Dose = (((Area + ySize*(i+5*RATIO) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else // i = xSize-1
        {
            Dose = (((Area + ySize*(xSize-1) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        TempPS = TempPS -> NextPS;
    }/* end while */


//check that the hyperdose sleave is small enough for Static Sources
    TempPS = FirstStaticPS;
    while (TempPS)
    {
        j = TempPS -> y;
        i = TempPS -> x;
        /* west */
        if (j>5*RATIO)
        {
            Dose = (((Area + ySize*i + (j-5*RATIO))->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else //j=0
        {
            Dose = (((Area + ySize*i)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* north */
        if (i>5*RATIO)
        {
            Dose = (((Area + ySize*(i-5*RATIO) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else //i=0
        {
            Dose = (((Area + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* east */
        if (j+5*RATIO<ySize)
        {
            Dose = (((Area + ySize*i + (j+5*RATIO))->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else //j=ySize-1
        {
            Dose = (((Area + ySize*i + (ySize-1))->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        /* south */
```

```
        if (i+5*RATIO<xSize)
        {
            Dose = (((Area + ySize*(i+5*RATIO) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        else // i = xSize-1
        {
            Dose = (((Area + ySize*(xSize-1) + j)->Dose)/Min);
            if (Dose>2.f)
                return(1);
        }
        TempPS = TempPS -> NextPS;
    }/* end while */


/*
//check that the hyperdose sleave is BIG enough
    TempPS = ThePS;
    while (TempPS)
    {
        j = TempPS -> y;
        i = TempPS -> x;
        // west
        if (j>4)
        {
            Dose = (((Area + ySize*i + (j-4))->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        else //j=0
        {
            Dose = (((Area + ySize*i)->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        // north
        if (i>4)
        {
            Dose = (((Area + ySize*(i-4) + j)->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        else //i=0
        {
            Dose = (((Area + j)->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        // east
        if (j+4<ySize)
        {
            Dose = (((Area + ySize*i + (j+4))->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        else //j=ySize-1
        {
            Dose = (((Area + ySize*i + (ySize-1))->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        // south
        if (i+4<xSize)
        {
            Dose = (((Area + ySize*(i+4) + j)->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        else // i = xSize-1
        {
```

```
            Dose = (((Area + ySize*(xSize-1) + j)->Dose)/Min);
            if (Dose<=2.f)
                return(-1);
        }
        TempPS = TempPS -> NextPS;
    }// end while
*/

// if nothing else has a problem, return that all is well (0)
    return(0);
}/*end function*/


/*************************************************************************/
/*                                  */
/*      Function : CalculateCost*/
/*  Purpose: This funciton will calculate the cost of the current*/
/*  solution space.             */
/*                                  */
/*************************************************************************/
float CalculateCost(PS *ThePS)
{
int i;
int j;
float Dose=0.0f;
PS *TempPS=NULL;
PS *TempStaticPS=NULL;
float MinTumour=9999999.f;
float MinPer = 9999999.f;
float MaxPer = 0.f;
int check1=0;
float TotalDoseToHyperCheckPoints=0;
int Count=0;
float A=0.f,B=0.f,C=0.f;

/*Go through the whole "Area" and calculate the dose*/
    for (i=0;i<xSize;i++)
    {
        for (j=0;j<ySize;j++)
        {
                Dose = 0.0f;
            // Run through Dynamic list of Sources
                TempPS = ThePS;
                while (TempPS)
                {
                    if ((TempPS -> x == i) && (TempPS -> y == j))
                        Dose += TempPS -> DwellTime * sqr(RATIO);
                    else
                        Dose += ((TempPS -> DwellTime)*sqr(RATIO))/sqr(distance(i,j,TempPS->x,TempPS-
>y));

                    TempPS = TempPS -> NextPS;
                }
            // Run through Static list of Sources
                TempStaticPS = FirstStaticPS;
                while (TempStaticPS)
                {
                    if ((TempStaticPS -> x == i) && (TempStaticPS -> y == j))
                        Dose += TempStaticPS -> DwellTime * sqr(RATIO);
                    else
                        Dose += ((TempStaticPS -> DwellTime)*sqr(RATIO))/sqr(distance(i,j,TempStaticPS-
>x,TempStaticPS->y));

                    TempStaticPS = TempStaticPS -> NextPS;
                }// end while
            // Store the Dose to the "Area" array
                (Area + ySize*i + j) -> Dose = Dose;

            /*might as well store the minimum to the tumour at the same time...*/
                if (((Area + ySize*i + j) -> Type == TUMOUR)&&(Dose<MinTumour))
                    MinTumour = Dose;
                else if (((Area + ySize*i + j) -> Type == PERIPHERY)&&(Dose<MinPer))
```

```
                MinPer = Dose;
            else if (((Area + ySize*i + j) -> Type == PERIPHERY)&&(Dose>MaxPer))
                MaxPer = Dose;
        }
    }


/* We have the minimum to the tumour and all of the doses calculated */
/* Now, we must find the hyper dose sleave in 4 dir'ns for ===DYNAMIC===*/
    TempPS = ThePS;
    while (TempPS)
    {
        j = TempPS -> y;
        i = TempPS -> x;
        /* west */
        if (j>5*RATIO)
        {
            Dose = (((Area + ySize*i + (j-5*RATIO))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        else //j=0
        {
            Dose = (((Area + ySize*i)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* north */
        if (i>5*RATIO)
        {
            Dose = (((Area + ySize*(i-5*RATIO) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        else //i=0
        {
            Dose = (((Area + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* east */
        if (j+5*RATIO<ySize)
        {
            Dose = (((Area + ySize*i + (j+5*RATIO))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        else //j=ySize-1
        {
            Dose = (((Area + ySize*i + (ySize-1))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* south */
        if (i+5*RATIO<xSize)
        {
            Dose = (((Area + ySize*(i+5*RATIO) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        else // i = xSize-1
        {
            Dose = (((Area + ySize*(xSize-1) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        TempPS = TempPS -> NextPS;
    }/* end while */


/* We have the minimum to the tumour and all of the doses calculated */
/* Now, we must find the hyper dose sleave in 4 dir'ns for ===STATIC===*/
    TempPS = FirstStaticPS;
    while (TempPS)
    {
        j = TempPS -> y;
        i = TempPS -> x;
```

```
        /* west */
        if (j>5*RATIO)
        {
            Dose = (((Area + ySize*i + (j-5*RATIO))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        else //j=0
        {
            Dose = (((Area + ySize*i)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* north */
        if (i>5*RATIO)
        {
            Dose = (((Area + ySize*(i-5*RATIO) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        else //i=0
        {
            Dose = (((Area + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* east */
        if (j+5*RATIO<ySize)
        {
            Dose = (((Area + ySize*i + (j+5*RATIO))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        else //j=ySize-1
        {
            Dose = (((Area + ySize*i + (ySize-1))->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        /* south */
        if (i+5*RATIO<xSize)
        {
            Dose = (((Area + ySize*(i+5*RATIO) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;}
        else // i = xSize-1
        {
            Dose = (((Area + ySize*(xSize-1) + j)->Dose)/MinPer);
            TotalDoseToHyperCheckPoints+=Dose;
            Count++;
        }
        TempPS = TempPS -> NextPS;
    }/* end while */
// Average hyperdose value
    A = ((float)TotalDoseToHyperCheckPoints/(float)Count)-2.f;
// Min to the tumour
    B = 1.f - MinTumour/MinPer;
// Max to periphery
    C = MaxPer/MinPer - 1.05f;

    if (B > 0.f)//min in the tumour is too cold
    {
        if (C > 0.f)//too much dose leaving the tumour
            return(A+B+C);
        else        //correct dose leaving tumour
            return(A+B);
    }// end if
    else// min in tumour OK
    {
        if (C > 0.f)//too much dose leaving the tumour
            return(A+C);
        else        //correct dose leaving tumour
            return(A);
    }// end else
    fprintf(GlobalOutputFile,"Major problem in cost function, exiting\n");
    exit(1);//there was a major error if it gets here
```

```
}/*end of function CalculateCost*/
```

E.15 CtrlNet.cpp

```
// CtrlNet.cpp: implementation of the CCtrlNet class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Isocurve.h"
#include "CtrlNet.h"

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

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CCtrlNet::CCtrlNet()
{
    m_Pos[0] = 0.0f;
    m_Pos[1] = 0.0f;
    m_Pos[2] = 0.0f;
}

CCtrlNet::~CCtrlNet()
{

}
```

E.16 CtrlNet.h

```
// CtrlNet.h: interface for the CCtrlNet class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_CTRLNET_H__ADCEACC7_A0F4_11D3_8081_0090276F4183__INCLUDED_)
#define AFX_CTRLNET_H__ADCEACC7_A0F4_11D3_8081_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "gl\gl.h"
#include "gl\glu.h"
#include "gl\glaux.h"

class CCtrlNet
{
public:
    GLfloat m_Pos[3];
    //CCtrlNet * m_pNextCtrlPt;
    CCtrlNet();
    virtual ~CCtrlNet();

};

#endif // !defined(AFX_CTRLNET_H__ADCEACC7_A0F4_11D3_8081_0090276F4183__INCLUDED_)
```

E.17 FileInputDlg.cpp

```
// FileInputDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "FileInputDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CFileInputDlg dialog


CFileInputDlg::CFileInputDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CFileInputDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CFileInputDlg)
    m_FileName = _T("");
    //}}AFX_DATA_INIT
}


void CFileInputDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFileInputDlg)
    DDX_Text(pDX, IDC_FILENAME, m_FileName);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CFileInputDlg, CDialog)
    //{{AFX_MSG_MAP(CFileInputDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFileInputDlg message handlers
```

E.18 FileInputDlg.h

```
#if !defined(AFX_FILEINPUTDLG_H__87B16540_4A52_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_FILEINPUTDLG_H__87B16540_4A52_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// FileInputDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CFileInputDlg dialog

class CFileInputDlg : public CDialog
{
// Construction
public:
    CFileInputDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
```

```
    //{{AFX_DATA(CFileInputDlg)
    enum { IDD = IDD_DIALOG1 };
    CStringm_FileName;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CFileInputDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CFileInputDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_FILEINPUTDLG_H__87B16540_4A52_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.19 Globals.h

```
/***********************************************/
/*                     */
/*  This header file contains definitions and  */
/*  variables needed in main.c for Simulated    */
/*  Annealing.     */
/*                     */
/***********************************************/
/*                     */
/*                     */
/*  CREATED: 26/08/98 - present*/
/*  AUTHOR : STEVEN MILLER*/
/*  Programmed for M.Sc. Thesis*/
/*  Version: 2.6.3.a*/
/*                     */
/***********************************************/



#ifndef _GLOBALS_H
#define _GLOBALS_H 1
#include <time.h>

#define sqr(x) ((x)*(x))

#define DEBUG_MOVE_SOURCES 0/* to debug the MoveSource Alg. */
#define DEBUG_CREATE_AREA 1/* to debug CreateAreaMap fnct */
#define DEBUG_INSERT_SOURCES 0/* to debug the insert source algorithm */
#define DEBUG_CHECK_COST 0/* to debug the check cost fnct. */
#define DEBUG_INIT_TEMP 0/* to debug the initial temp. fnct */
#define DEBUG_SIM_ANN 1/* to debug the main simulated annealing loop */
#define DEBUG_COST 0/* to debug the cost function */
#define DEBUG_PLOT_DOSE 0/* to debug the 3-D dose plotting function */
#define DEBUG_CHECKHYPERDOSE 0/* to debug the checkhyperdose function */


#define TUMOUR 0/* the grey scale value for pgm files */
#define PERIPHERY 40/* the grey level for the periphery of the tumour */
#define MOVABLE_LOCATION 80/* the grey scale value for locations which the source can move to */
```

```
#define POINT_SOURCE 175/* the grey scale of a point source for the pgm files */
#define EXTERNAL 255/* the grey scale value for pgm files */

#define PENALTY 10/* the penalty for point sources external to the tumour */



#define DEFAULT_TEMPERATURE 0.00001/* the default temperature for the simulation */
#define DEFAULT_STOP_COUNT 50/* this is the number of iterations that must have temp. within WITHIN */
#define DEFAULT_WITHIN_FACTOR 0.25/* the percentage that the cost can differ */
#define MAX_INIT_TEMP_ITERATIONS 100/* the number of iterations to get the initial temp */
#define INITIAL_ACCEPT_RANGE 0.005/* how close the curtemp must be to initial accept ratio to stop look-
ing */
#define INITIAL_ACCEPT_RATIO 0.95/* the acceptance ratio to start off with */

#define DEFAULT_NUMBER_OF_SOURCES 5/* the default number of sources to insert into the simulation */
#define DEFAULT_DWELL_TIME 100.0/* the default amount of time a source dwells for */
#define DEFAULT_PRESCRIBED_DOSE 15.8/* the default dose for the simulation */
#define DEFAULT_MAX_ITERATIONS 5000/* the number of iterations for each temperature value */
#define DISTANCE_PARAMETER 0.1/* used to calculate the new temperature based on the current */
#define MAX_SOURCE_MOVE 10/* the maximum number of mm that a source could move */
#define DEFAULT_TEMP_REDUCTION_FACTOR 0.9/* used to reduce the temperature of the simulation */
#define TEMP_INCREASE_FACTOR 1.5/* used to increase the temperature in the InitTemp function */
#define DEFAULT_INTERNAL_WEIGHT 1.0/* used to weight the contribution from internal dose */
#define DEFAULT_EXTERNAL_WEIGHT 1.0/* used to weight the contribution from external dose */
#define DEFAULT_EXTERNAL_FACTOR 0.25/* this is the percentage of prescribed dose that can be outside the
tumour*/
#define MOVE_PERCENT 0.98
#define ADD_PERCENT 0.01
#define DELETE_PERCENT 0.01
#define JUST_MOVE 1/* this is 1 if there is only move alowed, i.e. no insertions or deletions*/
#define ALLOW_0_SOURCES 0/* if this is 1 then it is OK to have 0 sources */
#define ALLOW_SOURCES_OUTSIDE 0/* if this is 1 then sources are not confined to the tumour */
#define TIME_FACTOR 0/* if this is 1 then there can be more than one source at a location */
#define USE_HEURISTIC 1/* usually this is set to 1 as well as JUST_MOVE so that the heuristic method is
used*/

typedef struct _DoseArea{
    float Dose;
    int Type;
    }DoseArea;

/* this is a structuure for the point sources */
typedef struct PointSource{
    int x;
    int y;
    float DwellTime;
    struct PointSource *NextPS;
    }PS;

extern float INTERNAL_WEIGHT;/* used to weight the contribution from internal dose */
extern float EXTERNAL_WEIGHT;/* used to weight the contribution from external dose */
extern float EXTERNAL_FACTOR;/* this is the percentage of prescribed dose that can be outside the
tumour*/

extern DoseArea *Area;/* This is the 2D array of structures */

extern int xSize;/* This is read in from the tumour input file */
extern int ySize;/* This is read in from the tumour input file */


extern PS *FirstPS;/* This is the linked list created in InsertSources */
extern PS *FirstNewPS;/* This is the linked list created from FirstPS in MoveSources */



extern float DWELL_TIME;/* global var. for dwell_time */
extern float PRESCRIBED_DOSE;/* the prescribed dose... */
extern float TEMP_REDUCTION_FACTOR;/* the amount to decrease temp after MAX_ITERATIONS */
extern int MAX_ITERATIONS;/* the max iterations in the sim. ann. b/f temp is reduced*/
extern int STOP_COUNT;/* if this many within WITHIN_FACTOR then stop */
extern float WITHIN_FACTOR;/* this is the % of iterations that have to be within */
```

```
extern int GlobalTumourCount;/* this is to count the number of tumour spots */
extern int HyperDose;/* This is the global variable to set if the dose is too high */


extern TempGlobal;

/*  FUNCTION PROTOTYPES, main.c*/
int round(float x);
float distance(int x1,int y1,int x2,int y2);
float GetInitialTemp(int NumSources);
void FillContoursRecursive(DoseArea *Array,int x,int y,int x_dim, int y_dim);
int PlotOutputDose(int Trial, PS **PSList);
int MakeOutputPGM(int Trial);
int CreateAreaMap(char Name[128]);
int SeedRandom(void);
int CheckCostFnct(void);
int main(int argc,char *argv[]);


/*  FUNCTION PROTOTYPES, PS.c */
void InsertSources(int NumSources);
int MoveSources(float InitialTemp,float CurTemp);
void FreePS(PS **list);
void SwapPS(PS **one, PS **two);
void CopyPS(PS **one, PS *two);
void PrintPS(PS *list);
void AddPS(PS *original_PS,PS **list);
void DeletePS(int WhichSource,PS *original_PS,PS **list);
void MakePS(PS **TempPS);
int MovePS(int WhichSource,PS *original_list,PS **new_list);
int NumPS(PS *TheList);

/*  FUNCTION PROTOTYPES, cost.c*/
int CheckHyperDoseSleave(PS *ThePS);
float CalculateCost(PS *TempPS);

/*  FUNCTION PROTOTYPES, sa.c*/
int SimulatedAnnealing(float InitialTemp, int Trial, int NumSources);

/*  FUNCTION PROTOTYPES, MyTime.c*/
void TotalTime(float Time);
void StartTime(time_t *lt);
void StopTime(time_t lt);
#endif
```

E.20 Isocurve.cpp

```
// Isocurve.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Isocurve.h"

#include "MainFrm.h"
#include "ChildFrm.h"
#include "IsocurveDoc.h"
#include "IsocurveView.h"
//#include "CoolingDoc.h"
//#include "CoolingView.h"


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

```
/////////////////////////////////////////////////////////////////////////////
// CIsocurveApp

BEGIN_MESSAGE_MAP(CIsocurveApp, CWinApp)
    //{{AFX_MSG_MAP(CIsocurveApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
    // Standard print setup command
    ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIsocurveApp construction

CIsocurveApp::CIsocurveApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CIsocurveApp object

CIsocurveApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CIsocurveApp initialization

BOOL CIsocurveApp::InitInstance()
{
    // Initialize OLE libraries
    if (!AfxOleInit())
    {
        AfxMessageBox(IDP_OLE_INIT_FAILED);
        return FALSE;
    }

    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

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

    // Change the registry key under which our settings are stored.
    // TODO: You should modify this string to be something appropriate
    // such as the name of your company or organization.
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings();  // Load standard INI file options (including MRU)

    // Register the application's document templates.  Document templates
    //  serve as the connection between documents, frame windows and views.

    CMultiDocTemplate* pDocTemplate;
    pDocTemplate = new CMultiDocTemplate(
        IDR_ISOCURTYPE,
        RUNTIME_CLASS(CIsocurveDoc),
        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
        RUNTIME_CLASS(CIsocurveView));
    pDocTemplate->SetContainerInfo(IDR_ISOCURTYPE_CNTR_IP);
```

```
    AddDocTemplate(pDocTemplate);// must call this every time we add a template!


/*  CMultiDocTemplate* p2DocTemplate;
    p2DocTemplate = new CMultiDocTemplate(
        IDR_COOLINGTYPE,
        RUNTIME_CLASS(CCoolingDoc),
        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
        RUNTIME_CLASS(CCoolingView));
    p2DocTemplate->SetContainerInfo(IDR_ISOCURTYPE_CNTR_IP);
    AddDocTemplate(p2DocTemplate);// must call this every time we add a template!
*/


    // create main MDI Frame window
    CMainFrame* pMainFrame = new CMainFrame;
    if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
        return FALSE;
    m_pMainWnd = pMainFrame;

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The main window has been initialized, so show and update it.
    pMainFrame->ShowWindow(m_nCmdShow);
    pMainFrame->UpdateWindow();

    return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
        // No message handlers
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
```

```
    //{{AFX_DATA_MAP(CAboutDlg)
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
        // No message handlers
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CIsocurveApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CIsocurveApp message handlers
```

E.21 Isocurve.h

```
// Isocurve.h : main header file for the ISOCURVE application
//

#if !defined(AFX_ISOCURVE_H__E8F0A3A4_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_ISOCURVE_H__E8F0A3A4_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_

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

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

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CIsocurveApp:
// See Isocurve.cpp for the implementation of this class
//

class CIsocurveApp : public CWinApp
{
public:
    CIsocurveApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CIsocurveApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CIsocurveApp)
    afx_msg void OnAppAbout();
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


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

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

#endif // !defined(AFX_ISOCURVE_H__E8F0A3A4_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.22 IsocurveDlg.cpp

```
// IsocurveDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "IsocurveDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CIsocurveDlg dialog


CIsocurveDlg::CIsocurveDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CIsocurveDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CIsocurveDlg)
    m_Isovalue = 0.0f;
    //}}AFX_DATA_INIT
}


void CIsocurveDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CIsocurveDlg)
    DDX_Text(pDX, IDC_EDIT1, m_Isovalue);
    DDV_MinMaxFloat(pDX, m_Isovalue, 1.e-002f, 9.f);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CIsocurveDlg, CDialog)
    //{{AFX_MSG_MAP(CIsocurveDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIsocurveDlg message handlers
```

E.23 IsocurveDlg.h

```
#if !defined(AFX_ISOCURVEDLG_H__D6E77700_48CF_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_ISOCURVEDLG_H__D6E77700_48CF_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// IsocurveDlg.h : header file
//
```

```
/////////////////////////////////////////////////////////////////////////////
// CIsocurveDlg dialog

class CIsocurveDlg : public CDialog
{
// Construction
public:
    CIsocurveDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CIsocurveDlg)
    enum { IDD = IDD_ISOCURVEDLG };
    floatm_Isovalue;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CIsocurveDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CIsocurveDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_ISOCURVEDLG_H__D6E77700_48CF_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.24 IsocurveDoc.cpp

```
// IsocurveDoc.cpp : implementation of the CIsocurveDoc class
//

#include "stdafx.h"
#include "Isocurve.h"
#include "IsocurveDoc.h"
#include "CntrItem.h"
#include "IsocurveDlg.h"
#include "FileInputDlg.h"
#include "StatsReportDlg.h"
#include "NewShapeDlg.h"
#include "AddPointSourceDlg.h"
#include "OutputPropsDlg.h"
#include "math.h"
#include "RatioDlg1.h"
#include "TextToSpeech.h"
#include "layer.h"
#include "multfactorDlg.h"

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

#define SQR(x) ((x)*(x))
/////////////////////////////////////////////////////////////////////////////
// CIsocurveDoc
```

```
IMPLEMENT_DYNCREATE(CIsocurveDoc, COleDocument)

BEGIN_MESSAGE_MAP(CIsocurveDoc, COleDocument)
    //{{AFX_MSG_MAP(CIsocurveDoc)
    ON_COMMAND(ID_FILE_NEW, OnFileNew)
    ON_COMMAND(ID_VIEW_ISOCURVE, OnViewIsocurve)
    ON_COMMAND(ID_VIEW_CLEARISOCURVE, OnViewClearisocurve)
    ON_COMMAND(ID_FILE_OUTPUT, OnFileOutput)
    ON_COMMAND(ID_FILE_INPUTPREDICTED, OnFileInputpredicted)
    ON_COMMAND(ID_VIEW_RATIO, OnViewRatio)
    ON_COMMAND(ID_VIEW_NEURALNETWORK, OnViewNeuralNetwork)
    ON_COMMAND(ID_VIEW_VIEWPREDICTIONS, OnViewViewpredictions)
    ON_COMMAND(ID_VIEW_COONSLINEAR, OnViewCoonslinear)
    ON_COMMAND(ID_VIEW_STATSREPORT, OnViewStatsreport)
    ON_COMMAND(ID_FILE_NEWSHAPE, OnFileNewshape)
    ON_COMMAND(ID_FILE_SAVE, OnFileSave)
    ON_COMMAND(ID_EDIT_ADDPOINTSOURCE, OnEditAddpointsource)
    ON_COMMAND(ID_EDIT_RESTORESOURCEPOSITIONS, OnEditRestoresourcepositions)
    ON_COMMAND(ID_EDIT_STOREENERGYASDESIREDOUTPUT, OnEditStoreEnergy)
    ON_COMMAND(ID_VIEW_MULTFACTFORCEV, OnViewMultfact)
    ON_COMMAND(ID_EDIT_CLEARALLPOINTSOURCES, OnEditClearPS)
    ON_COMMAND(ID_PRESCRIBE, OnPrescribe)
    ON_COMMAND(ID_SHOWHIDEISO, OnShowhideiso)
    ON_COMMAND(ID_VIEW_TOGGLEISOCURVE, OnViewToggleisocurve)
    ON_COMMAND(ID_FILE_SAVE_AS, OnFileSaveAs)
    ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
    //}}AFX_MSG_MAP
    // Enable default OLE container implementation
    ON_UPDATE_COMMAND_UI(ID_EDIT_PASTE, COleDocument::OnUpdatePasteMenu)
    ON_UPDATE_COMMAND_UI(ID_EDIT_PASTE_LINK, COleDocument::OnUpdatePasteLinkMenu)
    ON_UPDATE_COMMAND_UI(ID_OLE_EDIT_CONVERT, COleDocument::OnUpdateObjectVerbMenu)
    ON_COMMAND(ID_OLE_EDIT_CONVERT, COleDocument::OnEditConvert)
    ON_UPDATE_COMMAND_UI(ID_OLE_EDIT_LINKS, COleDocument::OnUpdateEditLinksMenu)
    ON_COMMAND(ID_OLE_EDIT_LINKS, COleDocument::OnEditLinks)
    ON_UPDATE_COMMAND_UI_RANGE(ID_OLE_VERB_FIRST, ID_OLE_VERB_LAST, COleDocument::OnUpdateObjectVerb-
Menu)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIsocurveDoc construction/destruction

CIsocurveDoc::CIsocurveDoc()
{
    // Use OLE compound files
    EnableCompoundFile();

    // TODO: add one-time construction code here
    m_ySize=0;
    m_xSize=0;
    m_Area=NULL;
    m_StoredArea=NULL;
    m_IsocurveSet = FALSE;
    m_FirstPS = NULL;
    m_FirstActualPS = NULL;
    m_FirstDesiredPS = NULL;
    m_FirstPredictedPS = NULL;
    m_Isovalue = 1.0f;
    m_FileName = "d:/steve/IsoCurve3/SourceConfig110.pgm";
    m_LoadedFile = 0;
    m_TumourCount = 0;
    m_Ratio = 1.0;
    m_ViewPredictions = FALSE;
    m_Coons = FALSE;
    m_NewIso = FALSE;
    m_StoredDesired = FALSE;
    m_MultFact = 1.9;
    SeedRandom();
}


CIsocurveDoc::~CIsocurveDoc()
```

```
{
    // delete the area map
    if (m_Area)
        delete [] m_Area;

    //delete stored area map
    if (m_StoredArea)
        delete [] m_StoredArea;

    // delete all point sources
    if (m_FirstPS)
    {
        PS * TempPS=NULL;

        TempPS = m_FirstPS;
        while (TempPS)
        {
            m_FirstPS = TempPS;
            TempPS = TempPS->NextPS;
            delete m_FirstPS;
        }
    }

    if (m_FirstActualPS)
    {
        PS * TempPS=NULL;

        TempPS = m_FirstActualPS;
        while (TempPS)
        {
            m_FirstActualPS = TempPS;
            TempPS = TempPS->NextPS;
            delete m_FirstActualPS;
        }
    }

    if (m_FirstDesiredPS)
    {
        PS * TempPS=NULL;

        TempPS = m_FirstDesiredPS;
        while (TempPS)
        {
            m_FirstDesiredPS = TempPS;
            TempPS = TempPS->NextPS;
            delete m_FirstDesiredPS;
        }
    }
}

BOOL CIsocurveDoc::OnNewDocument()
{
    if (!COleDocument::OnNewDocument())
        return FALSE;

    // TODO: add reinitialization code here
    // (SDI documents will reuse this document)

    return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CIsocurveDoc serialization

void CIsocurveDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
```

```
    else
    {
        // TODO: add loading code here
    }

    // Calling the base class COleDocument enables serialization
    //  of the container document's COleClientItem objects.
    COleDocument::Serialize(ar);
}

/////////////////////////////////////////////////////////////////////////////
// CIsocurveDoc diagnostics

#ifdef _DEBUG
void CIsocurveDoc::AssertValid() const
{
    COleDocument::AssertValid();
}

void CIsocurveDoc::Dump(CDumpContext& dc) const
{
    COleDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CIsocurveDoc commands

void CIsocurveDoc::OnFileNew()
{
    // OPEN FILE
    FILE *PGMFile=NULL;
    char name[128];
    int value=0;
    //PS *NewPS=NULL,*CurPS=NULL;
    PS *NewPS=NULL;

    //srgm12feb02 was this:
    /*
    CFileInputDlg dlg;
    dlg.m_FileName = CIsocurveDoc::m_FileName;
    if (dlg.DoModal()==IDOK)
        m_FileName = dlg.m_FileName;
    else
        return;
    */
    // now will be this:
    CFileDialog Dlg(TRUE,"*.pgm","*.pgm",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"PGM Files
(*.pgm)|*.pgm||");

    if (Dlg.DoModal() == IDOK)
    {
        m_FileName = Dlg.GetPathName();
    }
    else
        return;



    // get rid of previous run
    if (m_FirstPS)
    {
        NewPS = m_FirstPS;
        while (NewPS)
        {
            m_FirstPS = NewPS;
            NewPS = NewPS -> NextPS;
            free(m_FirstPS);
            m_FirstPS = NULL;
        }
    }
    DeleteAllPredictedPoints();
```

```
    PGMFile = fopen(m_FileName,"r");
    if (!PGMFile)
    {
        AfxMessageBox("File does not exist, can not open");
        m_LoadedFile = 0;
        return;
    }
    m_LoadedFile = 1;
    m_TumourCount = 0;
    OnViewClearisocurve();

    fgets(name,128,PGMFile);/* P2 */
    fgets(name,128,PGMFile);/* # CREATOR: XV Version 3.01  Rev: 3/30/93 */
    /* now get the x and y dimensions of the file */
    fscanf(PGMFile,"%d %d\n",&m_xSize,&m_ySize);

    fgets(name,128,PGMFile);/* 255 */
    m_Area = new DoseArea[m_xSize*m_ySize];

    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            fscanf(PGMFile,"%d",&value);
            m_Area[(i*m_ySize)+j].Type = value;
            // segment out the pointsources
            if (value == 175)
                AddPointSource(i,j,1);
            // count the valid tumour points
            if (value != 255)
                m_TumourCount++;
        }
    }
    fclose(PGMFile);

    // Calculate the dose
    CalculateDose();

    UpdateAllViews(NULL);
}

void CIsocurveDoc::OnViewIsocurve()
{
    if (!m_LoadedFile)
        AfxMessageBox("Please load a file first.");
    else if (m_FirstPS == NULL)
        AfxMessageBox("There are no point sources loaded, please create a point source, then plot an isocurve.");
    else
    {
        CIsocurveDlg dlg;
        dlg.m_Isovalue = CIsocurveDoc::m_Isovalue;
        if (dlg.DoModal()==IDOK)
        {
            m_Isovalue = dlg.m_Isovalue;
            m_IsocurveSet = TRUE;
            m_NewIso = TRUE;
            CalculateDose();
            UpdateAllViews(NULL);
        }
    }
}

void CIsocurveDoc::OnViewClearisocurve()
{
    m_IsocurveSet = FALSE;
    UpdateAllViews(NULL);
}

void CIsocurveDoc::CalculateDose()
{
    PS * TempPS;
```

```
    float Min=999999.9f;
    float MinMin=999999.9f;
    float Max=-100.0f;

// Clear dose array before recalculating
    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            m_Area[i*m_ySize + j].Dose = 0.0f;
        }
    }

// Run through area and find dose
    for (i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
                float Dose = 0.0f;
                TempPS = m_FirstPS;
                while (TempPS)
                {
                    // prevent div by 0 for on point source
                    if ((TempPS -> x == i) && (TempPS -> y == j))
                        //Dose += ((float)m_TumourCount*0.25f);
                        Dose += ((float)TempPS -> DwellTime)*2.0f;
                    else
                        Dose += (TempPS -> DwellTime*(float)m_Ratio)/SQR(Distance(i,j,TempPS->x,TempPS-
>y));

                    TempPS = TempPS -> NextPS;
                }

                m_Area[i*m_ySize + j].Dose = Dose;

                //might as well store the minimum to the periphery at the same time...
                if ((m_Area[m_ySize*i + j].Type == 40)&&(Dose<Min))
                {
                    Min = Dose;
                    m_MinX = i;
                    m_MinY = j;
                }

                // might as well store the max as well now
                if (Dose>Max)
                {
                    m_MaxX = i;
                    m_MaxY = j;
                    Max = Dose;
                }

                if (Dose<MinMin)
                {
                    m_MinOutsideX = i;
                    m_MinOutsideY = j;
                    MinMin = Dose;
                }
        }
    }
// Now turn into percentages
    if (Min>0.000001)
    {
        for (i=0;i<m_xSize;i++)
        {
            for (int j=0;j<m_ySize;j++)
                m_Area[m_ySize*i + j].Dose = m_Area[m_ySize*i + j].Dose/(float)Min;
        }
    }
}

float CIsocurveDoc::Distance(int x1, int y1, int x2, int y2)
{
```

```
    if ((x1-x2==0) && (y1-y2==0))
        return(0.0);
    else
        return((float)sqrt(SQR(x1-x2)+SQR(y1-y2)));
}

void CIsocurveDoc::AddPointSource(int x,int y,float DwellTime)
{
    PS * NewPS=NULL;

    // make new one first
    NewPS = new (PS);
    NewPS->x = x;
    NewPS->y = y;
    NewPS->DwellTime = DwellTime;
    NewPS->NextPS = NULL;

    // now add it
    if (!m_FirstPS)
        m_FirstPS = NewPS;
    else
    {
        PS * LastPS=NULL;
        PS * SecLastPS=NULL;
        LastPS = m_FirstPS;
        // get to end of list
        while (LastPS)
        {
            SecLastPS = LastPS;
            LastPS = LastPS -> NextPS;
        }
        //SecLastPS is the last PS
        SecLastPS->NextPS = NewPS;
    }
}

void CIsocurveDoc::DeletePointSource(int i, int j)
{
    PS * CurPS=NULL;
    PS * PrevPS=NULL;
    int WasTheFirst = 0;

    CurPS = m_FirstPS;
    PrevPS = CurPS;
    while (CurPS)
    {
        if ((CurPS->x == i) && (CurPS->y == j))
        {
            PrevPS -> NextPS = CurPS -> NextPS;
            if (CurPS == m_FirstPS)
            {
                PrevPS = CurPS-> NextPS;
                m_FirstPS = CurPS->NextPS;
                WasTheFirst = 1;
            }
            delete CurPS;
            CurPS = PrevPS;
        }
        PrevPS = CurPS;
        if (CurPS)
            CurPS = CurPS->NextPS;
    }

    // if we deleted the last PS then unset IsocurveSet
    if (WasTheFirst && !m_FirstPS)
        m_IsocurveSet = FALSE;
}

void CIsocurveDoc::OnFileOutput()
{
    // check that a file is loaded
    if (!m_Area)
```

```
    {
        AfxMessageBox("Please load a tumour file first\n");
        return;
    }

    // prompt user for output paramaters
    static int Manhattan=0;
    static int NumInst=1;
    COutputPropsDlg dlg;
    dlg.m_Manhattan = Manhattan;
    dlg.m_NumInst = NumInst;
    if (dlg.DoModal() == IDOK)
    {
        static FILE *OutputFile = NULL;
        if (!dlg.m_New) //create new file
        {
            // srgm12feb02 was this:
            /*
            OutputFile = fopen("d:/steve/msc/isocurve3/training.dat","w");
            */
            // now this:
            CFileDialog fDlg(TRUE,"*.dat","*.dat",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"PGM Files
(*.pgm)|*.pgm||");
            if (fDlg.DoModal() == IDOK)
            {
                //m_FileName = fDlg.GetPathName();
                OutputFile = fopen(fDlg.GetPathName(),"w");
            }
            else
                return;
        }

        // srgm12feb02 was this:
        /*
        if (!OutputFile)//maybe they just forgot to open it so lets be nice and open one
            OutputFile = fopen("d:/steve/msc/isocurve3/training.dat","w");
        */
        //now this:
        if (!OutputFile)
        {
            AfxMessageBox("There is no file open for writting output!");
            return;
        }

        //make sure that it opened
        if (!OutputFile)
        {
            AfxMessageBox("File training.dat was not opened");
            return;
        }

        NumInst = dlg.m_NumInst;
        Manhattan = dlg.m_Manhattan;
        for (int i=0;i<dlg.m_NumInst;i++)
        {
            //call function to play with source config
            StoreSources();//store the sources
            MoveSourcesAround(dlg.m_Manhattan);//manhatan distance! not euclidean
            CalculateDose();// make sure that the dose is calculated
            OnEditRestoresourcepositions();//have to call this to delete the stored sources

            //find max in the Area to divide energy function by
            float maxArea = m_Area[0].Dose;
            for (int i=0;i<m_xSize;i++)
            {
                for (int j=0;j<m_ySize;j++)
                {
                    if ((m_Area[(i*m_ySize)+j].Type != 255) && (m_Area[(i*m_ySize)+j].Dose > maxArea))
                        maxArea = m_Area[(i*m_ySize)+j].Dose;
                }
            }
```

```
            // this is the inputs for training
            for (i=0;i<m_xSize;i++)
            {
                for (int j=0;j<m_ySize;j++)
                {
                    if (m_Area[(i*m_ySize)+j].Type != 255)
                        if (maxArea > 0.000001)
                            fprintf(OutputFile,"%f ",((m_Area[(i*m_ySize)+j].Dose)/(float)maxArea));
                        else
                            fprintf(OutputFile,"0.0 ");
                }
            }
            //make sure to include the ratio
            fprintf(OutputFile,"%f ",m_Ratio);

            //find max in the StoredArea to divide energy function by
            float maxStoredArea = m_StoredArea[0].Dose;
            for (i=0;i<m_xSize;i++)
            {
                for (int j=0;j<m_ySize;j++)
                {
                    if ((m_StoredArea[(i*m_ySize)+j].Type != 255) && (m_StoredArea[(i*m_ySize)+j].Dose >
maxStoredArea))
                        maxStoredArea = m_StoredArea[(i*m_ySize)+j].Dose;
                }
            }

            //desired outputs, depends on continuous or descrete
            if (!dlg.m_Continous) //continuous stuff
            {
                for (i=0;i<m_xSize;i++)
                {
                    for (int j=0;j<m_ySize;j++)
                    {
                        if (m_StoredArea[(i*m_ySize)+j].Type != 255)
                            if (maxStoredArea > 0.000001)
                                fprintf(OutputFile,"%f ",((m_StoredArea[(i*m_ySize)+j].Dose)/(float)max-
StoredArea));
                            else
                                fprintf(OutputFile,"0.0 ");
                    }//end for
                }//end for
            }
            else// else descrete stuff
            {
                for (i=0;i<m_xSize;i++)
                {
                    for (int j=0;j<m_ySize;j++)
                    {
                        if (m_StoredArea[(i*m_ySize)+j].Type != 255)
                        {
                            if (m_StoredArea[(i*m_ySize)+j].Type == 175)
                                fprintf(OutputFile,"1.0 ");
                            else
                                fprintf(OutputFile,"0.0 ");
                        }//end if
                    }//end for
                }//end for
            }// end else
            fprintf(OutputFile,"\n");//advance file pointer to next line
        }//end for

        if ((i==dlg.m_NumInst-1) && (dlg.m_New == 2) && (OutputFile))//close file
            fclose(OutputFile);
    }//end if
}//end function OnFileOutput

void CIsocurveDoc::AddDesiredPointSource(int x, int y,float DwellTime)
{
    PS * NewPS=NULL;

    // make new one first
```

```
    NewPS = new (PS);
    NewPS->x = x;
    NewPS->y = y;
    NewPS->DwellTime = DwellTime;
    NewPS->NextPS = NULL;

    // now add it
    if (!m_FirstDesiredPS)
        m_FirstDesiredPS = NewPS;
    else
    {
        PS * LastPS=NULL;
        PS * SecLastPS=NULL;
        LastPS = m_FirstDesiredPS;
        // get to end of list
        while (LastPS)
        {
            SecLastPS = LastPS;
            LastPS = LastPS -> NextPS;
        }
        //SecLastPS is the last PS
        SecLastPS->NextPS = NewPS;
    }
}

void CIsocurveDoc::OnFileInputpredicted()
{
    /*FILE * p_File;

    if (p_File = fopen("d:/steve/isocurve3/nn_Pred.dat","r"))
    {
        // get rid of current sources...
        if (m_FirstPS)
        {
            PS * TempPS=NULL;

            while (m_FirstPS)
            {
                TempPS = m_FirstPS;
                m_FirstPS = m_FirstPS->NextPS;
                m_Area[TempPS->x*m_ySize + TempPS->y].Type = 0;
                delete TempPS;
            }
        }
        // add predicted sources...

        if (!m_FirstPS)
            AfxMessageBox("There is no first point source");

        float value=0.0f;
        for (int i=0;i<m_xSize;i++)
        {
            for (int j=0;j<m_ySize;j++)
            {
                fscanf(p_File,"%f",&value);
                if (value == 1.0)
                {
                    AfxMessageBox("Adding");
                    m_Area[i*m_ySize+j].Type = 175;
                    AddPointSource(i,j,1);
                }
            }
        }
        UpdateAllViews(NULL);
    }
    else
        AfxMessageBox("There was a problem opening the predicted file\n");*/
    AfxMessageBox("This function has been temporarily turned off.");
}

void CIsocurveDoc::OnViewRatio()
{
```

```
    CRatioDlg1 dlg;

    dlg.m_Ratio = m_Ratio;
    if (dlg.DoModal() == IDOK)
        m_Ratio = dlg.m_Ratio;
}

void CIsocurveDoc::OnViewNeuralNetwork()
{
    CTextToSpeech * pTTS = new CTextToSpeech;


    //delete current predicted layout
    DeleteAllPredictedPoints();

    // pass in current energy function  to NN - using current source configuration
    if (!m_Area)
    {
        pTTS->SpeakText("Please load a tumor file for analysis first.");
        return;
    }
    else
        pTTS->SpeakText("Neural network is predicting where to place sources.  One moment please.");

    //if (m_FirstPS)
    //  pTTS->SpeakText("there are point sources");

    CalculateDose();// make sure that the dose is calculated

    FILE * OutputFile;
    if ((OutputFile=fopen("nninput.dat","w"))==NULL)
    {
        pTTS->SpeakText("Problem opening neural network input file for writing, exiting Neural Network
algorithm");
        return;
    }

    float max = m_Area[0].Dose;
    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            if ((m_Area[(i*m_ySize)+j].Type != 255) && (m_Area[(i*m_ySize)+j].Dose > max))
                max = m_Area[(i*m_ySize)+j].Dose;
        }
    }

// this is the current energy function
    for (i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            if (m_Area[(i*m_ySize)+j].Type != 255)
            {
                if (max>0.00001)
                    fprintf(OutputFile,"%f ",((m_Area[(i*m_ySize)+j].Dose)/(float)max));
                else// divide by 0, so just as a default put out 0.0s
                    fprintf(OutputFile,"0.0 ");
            }
        }
    }
// make sure to add scaling factor
    fprintf(OutputFile,"%f\n",m_Ratio);
    fclose(OutputFile);

    // setup the BPNN
    network BPNN;
    BPNN.set_training(0);// not training
    int NetLayOut[10] = {901,12,900,0,0,0,0,0,0,0};
    // layer sizes
    BPNN.set_layer_info(3,NetLayOut);
    // make interconnections
```

```
BPNN.set_up_network();

// set weights
FILE * weights_file_ptr, * data_file_ptr, * output_file_ptr;
if ((weights_file_ptr=fopen("weights.dat","r"))==NULL)
{
    pTTS->SpeakText("Problem opening weights file, exiting Neural Network algorithm.");
    return;
}
BPNN.read_weights(weights_file_ptr);

if ((data_file_ptr=fopen("nninput.dat","r"))==NULL)
{
    pTTS->SpeakText("problem opening training file, exiting Neural network algorithm.");
    return;
}
BPNN.fill_IObuffer(data_file_ptr);

BPNN.set_up_pattern(0);// there is only the one
BPNN.forward_prop();
if ((output_file_ptr=fopen("output.dat","w"))==NULL)
{
    pTTS->SpeakText("Problem opening nn output file, exiting Neural Network algorithm");
    return;
}
BPNN.write_outputs(output_file_ptr,NULL);
fclose(output_file_ptr);
fclose(data_file_ptr);
fclose(weights_file_ptr);


// Going to try and see what happens if we subtract current energy function for NN output
// then doing thresholding
//retrive output and high pass filter it for desired locations

//NN output
if ((output_file_ptr=fopen("output.dat","r"))==NULL)
{
    pTTS->SpeakText("Problem opening nn output input file, exiting Neural Network algorithm");
    return;
}
//Current energy function
if ((data_file_ptr=fopen("nninput.dat","r"))==NULL)
{
    pTTS->SpeakText("Problem opening nn output input file, exiting Neural Network algorithm");
    return;
}

float value=0.0f;
float cev=0.0f;// current energy value
m_PredMax = -99999.0f;
for (i=0;i<m_xSize;i++)
{
    for (int j=0;j<m_ySize;j++)
    {
        fscanf(output_file_ptr,"%f",&value);//nn output
        fscanf(data_file_ptr,"%f",&cev);//nn input
        if ((value-(cev*m_MultFact)) > m_PredMax)
            m_PredMax = value;
    }
}
fclose(output_file_ptr);
fclose(data_file_ptr);

if ((output_file_ptr=fopen("output.dat","r"))==NULL)
{
    pTTS->SpeakText("Problem opening nn output input file, exiting Neural Network algorithm");
    return;
}
//Current energy function
if ((data_file_ptr=fopen("nninput.dat","r"))==NULL)
{
```

```
        pTTS->SpeakText("Problem opening nn output input file, exiting Neural Network algorithm");
        return;
    }


    // clear all predictions
    for (i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            m_Area[((i)*m_ySize)+(j)].Pred = 0.0f;
        }
    }

    // now calculate
    for (i=4;i<=m_xSize-7;i++)
    {
        for (int j=4;j<=m_ySize-7;j++)
        {
            fscanf(output_file_ptr,"%f",&value);
            fscanf(data_file_ptr,"%f",&cev);
            //if ((value-cev) > max*0.4)
            //{
                m_Area[((i)*m_ySize)+(j)].Pred = value-(cev*m_MultFact);
                if (m_Area[((i)*m_ySize)+(j)].Pred<m_PredMax*0.4)
                    m_Area[((i)*m_ySize)+(j)].Pred = 0.0f;

                //AddPredictedPoint(i+4,j+4,value);
            //}
        }
    }
    fclose(output_file_ptr);
    fclose(data_file_ptr);


    //set predicted point source list according to output

    //redraw sceen
    m_ViewPredictions = TRUE;
    UpdateAllViews(NULL);

    delete pTTS;
}

void CIsocurveDoc::OnViewViewpredictions()
{
    if (m_ViewPredictions)
        m_ViewPredictions = FALSE;
    else
        m_ViewPredictions = TRUE;
    UpdateAllViews(NULL);
}

void CIsocurveDoc::AddPredictedPoint(int x, int y,float DwellTime)
{
    PS * NewPS=NULL;

    // make new one first
    NewPS = new (PS);
    NewPS->x = x;
    NewPS->y = y;
    NewPS->DwellTime = DwellTime;
    NewPS->NextPS = NULL;

    // now add it
    if (!m_FirstPredictedPS)
        m_FirstPredictedPS = NewPS;
    else
    {
        PS * LastPS=NULL;
        PS * SecLastPS=NULL;
        LastPS = m_FirstPredictedPS;
```

```
        // get to end of list
        while (LastPS)
        {
            SecLastPS = LastPS;
            LastPS = LastPS -> NextPS;
        }
        //SecLastPS is the last PS
        SecLastPS->NextPS = NewPS;
    }
}

void CIsocurveDoc::DeleteAllPredictedPoints()
{
    PS * CurPS=NULL;
    PS * NextPS=NULL;

    CurPS = m_FirstPredictedPS;
    while (CurPS)
    {
        NextPS = CurPS->NextPS;
        delete CurPS;
        CurPS = NextPS;
    }
    m_FirstPredictedPS = NULL;
}

void CIsocurveDoc::OnViewCoonslinear()
{
    if (!m_Coons)
        m_Coons = TRUE;
    else
        m_Coons = FALSE;

    m_NewIso = TRUE;
    UpdateAllViews(NULL);
}

void CIsocurveDoc::OnViewStatsreport()
{
    if (!m_LoadedFile)
        AfxMessageBox("Please load a file first.");
    else
    {
        CStatsReportDlg dlg;
        dlg.m_Homo = 0.0f;
        dlg.m_Hyp = 0.0f;
        dlg.m_PerCheck = 0.0f;
        dlg.m_Fitness = 0.0f;
        GenerateStats();
        dlg.m_Homo = m_HomoIndex;
        dlg.m_Hyp = m_HyperIndex;
        dlg.m_PerCheck = m_MaxVSMinIndex;
        if (m_HomoIndex > 0.f)//min in the tumour is too cold
        {
            if (m_MaxVSMinIndex > 0.f)//too much dose leaving the tumour
                dlg.m_Fitness = m_HyperIndex+m_HomoIndex+m_MaxVSMinIndex;
            else        //correct dose leaving tumour
                dlg.m_Fitness = m_HyperIndex+m_HomoIndex;
        }// end if
        else// min in tumour OK
        {
            if (m_MaxVSMinIndex > 0.f)//too much dose leaving the tumour
                dlg.m_Fitness = m_HyperIndex+m_MaxVSMinIndex;
            else        //correct dose leaving tumour
                dlg.m_Fitness = m_HyperIndex;
        }// end else
        dlg.DoModal();
    }

}

void CIsocurveDoc::GenerateStats()
```

```
{
    float MinPer=999999.9f;
    float MinTum=999999.9f;
    float MaxPer=-100.0f;
    DoseArea *tArea;
    PS * TempPS=NULL;

    tArea = new DoseArea[m_xSize*m_ySize];

// Clear dose array before recalculating
    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            tArea[i*m_ySize + j].Dose = 0.0f;
            tArea[i*m_ySize + j].Type = m_Area[i*m_ySize + j].Type;
        }
    }

// Run through area and find dose
    for (i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
                float Dose = 0.0f;
                TempPS = m_FirstPS;
                while (TempPS)
                {
                    // prevent div by 0 for on point source
                    if ((TempPS -> x == i) && (TempPS -> y == j))
                        //Dose += ((float)m_TumourCount*0.25f);
                        Dose += ((float)TempPS -> DwellTime)*2.0f;
                    else
                        Dose += (TempPS -> DwellTime*(float)m_Ratio)/SQR(Distance(i,j,TempPS->x,TempPS-
>y));

                    TempPS = TempPS -> NextPS;
                }
                //(m_Area + m_ySize*i + j) -> Dose = Dose;
                tArea[i*m_ySize + j].Dose = Dose;

                //might as well store the minimum to the periphery at the same time...
                if ((tArea[m_ySize*i + j].Type == 40)&&(Dose<MinPer))
                    MinPer = Dose;

                // might as well store the max to the periphery as well now
                if ((tArea[m_ySize*i + j].Type == 40)&&(Dose>MaxPer))
                    MaxPer = Dose;

                // also need to store min to tumour itself
                if ((tArea[m_ySize*i + j].Type == 0)&&(Dose<MinTum))
                    MinTum = Dose;
        }
    }

//check that the hyperdose sleave is small enough for Dynamic Sources
    int RATIO;

    RATIO = (int)1/(m_Ratio+0.01);
    // turn ration into integer:

    // addup hyperdose stuff
    TempPS = m_FirstPS;
    int check=0;
    float Dose=0.0f;
    while (TempPS)
    {
        check++;
        i = TempPS -> x;
        int j = TempPS -> y;
        /* west */
        if (i>5*RATIO)
```

```
            Dose += tArea[(i-5*RATIO)*m_ySize + j].Dose;
        else //i=0
            Dose += tArea[j].Dose;
        /* north */
        if (j>5*RATIO)
            Dose += tArea[(i)*m_ySize + j-5*RATIO].Dose;
        else //j=0
            Dose += tArea[(i)*m_ySize].Dose;

        /* east */
        if (i+5*RATIO<m_xSize)
            Dose += tArea[(i+5*RATIO)*m_ySize+j].Dose;
        else //j=ySize-1
            Dose += tArea[(m_xSize-1)*m_ySize+j].Dose;
        /* south */
        if (j+5*RATIO<m_ySize)
            Dose += tArea[i*m_ySize+j+5*RATIO].Dose;
        else // i = xSize-1
            Dose += tArea[i*m_ySize+m_ySize-1].Dose;

        TempPS = TempPS -> NextPS;
    }/* end while */

    m_HomoIndex = 1.0f - (float)MinTum/(float)MinPer;
    m_MaxVSMinIndex = (float)MaxPer/(float)MinPer - 1.05f;
    m_HyperIndex = (Dose/(check*4*MinPer))-2.0f;

    if (tArea)
        delete [] tArea;
}//end function GenerateStats

void CIsocurveDoc::OnFileNewshape()
{
    CNewShapeDlg dlg;
    BOOL SizeOK=FALSE;

    dlg.m_CanvasX = 40;
    dlg.m_CanvasY = 40;
    dlg.m_SquareX = 10;
    dlg.m_SquareY = 10;
    dlg.m_CrclRad = 5;

    int done=0;
    while (!done)
    {
        dlg.m_Circle = 0;
        if (dlg.DoModal()== IDOK)
        {
            if (dlg.m_Circle)
            {
                if ((dlg.m_CrclRad*2<dlg.m_CanvasX) && (dlg.m_CrclRad*2<dlg.m_CanvasY))
                    SizeOK = TRUE;
            }
            else
            {
                if ((dlg.m_CanvasX>dlg.m_SquareX) && (dlg.m_CanvasY>dlg.m_SquareY))
                    SizeOK = TRUE;
            }

            if (SizeOK)
            {
                done = 1;
                //clear the current shape if any
                if (m_Area)
                    delete [] m_Area;

                // delete all point sources
                if (m_FirstPS)
                {
                    PS * TempPS=NULL;

                    TempPS = m_FirstPS;
```

```
                while (TempPS)
                {
                    m_FirstPS = TempPS;
                    TempPS = TempPS->NextPS;
                    delete m_FirstPS;
                }
            }
            m_FirstPS = NULL;

            //delete desired PS list (which should no longer exist)
            if (m_FirstDesiredPS)
            {
                PS * TempPS=NULL;

                TempPS = m_FirstDesiredPS;
                while (TempPS)
                {
                    m_FirstDesiredPS = TempPS;
                    TempPS = TempPS->NextPS;
                    delete m_FirstDesiredPS;
                }
            }
            m_FirstDesiredPS = NULL;

            //create new area
            m_xSize = dlg.m_CanvasX;
            m_ySize = dlg.m_CanvasY;
            m_Area = new DoseArea[m_xSize*m_ySize];

            if (dlg.m_Circle)
                MakeCircle(dlg.m_CrclRad);
            else
                MakeSquare(dlg.m_SquareX,dlg.m_SquareY);
        }//if SizeOK
        else
            AfxMessageBox("Sorry there are problems with the Canvas and shape sizes please re-enter
them.");
    }//end if dlg.domodal
    else
        return;
}//end while
}

void CIsocurveDoc::MakeCircle(int radius)
{
    m_LoadedFile = 1;
    m_TumourCount = 0;
    OnViewClearisocurve();

    //find offsets/center
    int xoff=0,yoff=0;

    xoff = m_xSize/2;
    yoff = m_ySize/2;

    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            int x,y;
            x = xoff-i;
            y = yoff-j;
            if (x*x+y*y<(radius-1)*(radius-1))
                m_Area[(i*m_ySize)+j].Type = 0;
            else if (x*x+y*y<radius*radius)
                m_Area[(i*m_ySize)+j].Type = 40;
            else
                m_Area[(i*m_ySize)+j].Type = 255;
        }
    }
    UpdateAllViews(NULL);
}
```

```
void CIsocurveDoc::MakeSquare(int x, int y)
{
    m_LoadedFile = 1;
    m_TumourCount = 0;
    OnViewClearisocurve();

    //find offsets/center
    int xoff=0,yoff=0;

    xoff = (m_xSize - x)/2;
    yoff = (m_ySize - y)/2;

    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            int xtype,ytype,type;

            if (i==xoff+1)
                xtype = 40;
            else if ((i>xoff+1) && (i<xoff+x))
                xtype = 0;
            else if (i==xoff+x)
                xtype = 40;
            else
                xtype = 255;

            if (j==yoff+1)
                ytype = 40;
            else if ((j>yoff+1) && (j<yoff+y))
                ytype = 0;
            else if (j==yoff+y)
                ytype = 40;
            else
                ytype = 255;

            //combine the types
            if (xtype==255 || ytype==255)
                type=255;
            else if (xtype==40 || ytype==40)
                type=40;
            else
                type=0;

            m_Area[(i*m_ySize)+j].Type = type;
        }
    }
    UpdateAllViews(NULL);
}

void CIsocurveDoc::OnFileSave()
{
    CFileInputDlg dlg;

    dlg.m_FileName = "d:/steve/MSc/Isocurve3/test.pgm";
    if (dlg.DoModal()==IDOK)
        m_FileName = dlg.m_FileName;
    else
        return;

    FILE * OutputFile=NULL;
    OutputFile = fopen(m_FileName,"w");

    fprintf(OutputFile,"P2\n");/* P2 */
    fprintf(OutputFile,"# CREATOR: Steve Miller, 1999\n");/* # CREATOR: XV Version 3.01  Rev: 3/30/93 */
    /* now put the x and y dimensions of the file */
    fprintf(OutputFile,"%d %d\n",m_xSize,m_ySize);
    // number of colors
    fprintf(OutputFile,"255\n");/* 255 */

    int counter=0;
```

```
    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            fprintf(OutputFile,"%d ",m_Area[(i*m_ySize)+j].Type);
            counter++;
            if (counter==17)
            {
                fprintf(OutputFile,"\n");
                counter=0;
            }
        }
    }


    fclose(OutputFile);
}

void CIsocurveDoc::OnEditAddpointsource()
{
    //Just bring up a Dlg box and create a new point source at the specified location
    CAddPointSourceDlg dlg;

    dlg.m_X = 0;
    dlg.m_Y = 0;
    dlg.m_DwellTime = 1;
    if (dlg.DoModal() == IDOK)
    {
        if (dlg.m_X<m_xSize && dlg.m_Y<m_ySize)//make sure it is in range
        {
            if (m_Area[dlg.m_X*m_ySize+dlg.m_Y].Type != 175)//make sure there is not one there
            {
                AddPointSource(dlg.m_X,dlg.m_Y,(float)dlg.m_DwellTime);
                m_Area[dlg.m_X*m_ySize+dlg.m_Y].Type = 175;
                UpdateAllViews(NULL);
            }
            else
                AfxMessageBox("There is already a source at that location, can not add another");
        }
        else
            AfxMessageBox("Sorry but the point source you entered was out of range");
    }
}

void CIsocurveDoc::MoveSourcesAround(int Distance)
{
    CString s;
    static int FirstTime=1;
    PS * NewPS=NULL;
    PS * TempPS=NULL;

    // to get rid of the global variables for number of sources:
    int NumSources = NumPS(m_FirstPS);
    if (NumSources==0)
        return;

    /* this is for deciding wether to move or add */
    float TempRand = (float)rand()/RAND_MAX;
    int WhichSource=0;

    for (int i=0;i<NumSources;i++)
    {
        if ((((float)(i)/NumSources) < TempRand) && (TempRand < (float)(i+1)/NumSources))
            WhichSource = i;
    }

    s.Format("Moving source: %d",WhichSource);
    //AfxMessageBox(s);
    MovePS(WhichSource,Distance);
}

void CIsocurveDoc::OnEditRestoresourcepositions()
```

```
{
    PS * TempPS,* TempPS2;
    //set FirstPS back to stored
    TempPS = m_FirstActualPS;
    TempPS2 = m_FirstPS;
    while (TempPS)
    {
        m_Area[TempPS2->x*m_ySize+TempPS2->y].Type = 0;
        TempPS2->x = TempPS->x;
        TempPS2->y = TempPS->y;
        TempPS2->DwellTime = TempPS->DwellTime;
        m_Area[TempPS2->x*m_ySize+TempPS2->y].Type = 175;

        TempPS2 = TempPS2->NextPS;
        TempPS = TempPS -> NextPS;
    }//end while

    //delete stored
    if (m_FirstActualPS)
    {
        PS * TempPS=NULL;

        TempPS = m_FirstActualPS;
        while (TempPS)
        {
            m_FirstActualPS = TempPS;
            TempPS = TempPS->NextPS;
            delete m_FirstActualPS;
        }
    }
    m_FirstActualPS=NULL;
    //UpdateAllViews(NULL);
}


void CIsocurveDoc::MovePS(int WhichSource,int Distance)
{
    int xMove=0,yMove=0;
    // this is to find how far to move in x direction
    float TempRand = (float)rand()/RAND_MAX;
    for (int i=0;i<Distance;i++)
    {
        if ((((float)(i)/Distance) < TempRand) && (TempRand < (float)(i+1)/Distance))
        {
            xMove = i;
            break;
        }
    }

    // this is to find how far to move in y direction
    TempRand = (float)rand()/RAND_MAX;
    for (i=0;i<Distance;i++)
    {
        if ((((float)(i)/Distance) < TempRand) && (TempRand < (float)(i+1)/Distance))
        {
            yMove = i;
            break;
        }
    }

    //make sure distance is OK
    if (xMove+yMove>Distance)
    {
        TempRand = (float)rand()/RAND_MAX;
        if (TempRand>0.5)//delete xMove
            xMove = 0;
        else
            yMove = 0;

    }

    //should x be Neg?  if Rand>.5 yes
```

```
    TempRand = (float)rand()/RAND_MAX;
    if (TempRand>0.5f)
        xMove= xMove*-1.0;
    //should y be Neg?  if Rand>.5 yes
    TempRand = (float)rand()/RAND_MAX;
    if (TempRand>0.5f)
        yMove= yMove*-1.0;

    /*CString s;
    s.Format("%d %d",xMove,yMove);
    AfxMessageBox(s);*/

    PS *TempPS=m_FirstPS;
    i=0;
    while (TempPS)
    {
        if (i==WhichSource)
        {
            if ((m_Area[(TempPS->x+xMove)*m_ySize+TempPS->y+yMove].Type == 0) ||
                (m_Area[(TempPS->x+xMove)*m_ySize+TempPS->y+yMove].Type == 40))
            {
                m_Area[TempPS->x*m_ySize+TempPS->y].Type = 0;
                TempPS->x+=xMove;
                TempPS->y+=yMove;
                m_Area[TempPS->x*m_ySize+TempPS->y].Type = 175;
            }
            break;
        }
        i++;
        TempPS = TempPS->NextPS;
    }
}//end function MovePS

int CIsocurveDoc::NumPS(PS *ThePSList)
{
    int counter=0;
    PS *TempPS=NULL;

    TempPS = ThePSList;
    while (TempPS)
    {
        counter++;
        TempPS = TempPS -> NextPS;
    }
    return(counter);
}

void CIsocurveDoc::SeedRandom()
{
        long ltime=0;
        int stime=0;

        /* get current calender time */
        ltime = time(NULL);
        stime = (unsigned) ltime/2;
        srand(stime);
        //end seed random
}

void CIsocurveDoc::StoreSources()
{
    PS * TempPS=NULL,* NewPS=NULL;

    TempPS = m_FirstPS;
    while (TempPS)
    {
        NewPS = new (PS);

        NewPS->x = TempPS->x;
        NewPS->y = TempPS->y;
        NewPS->DwellTime = TempPS->DwellTime;
        NewPS->NextPS = NULL;
```

```
        // now add it
        if (!m_FirstActualPS)
            m_FirstActualPS = NewPS;
        else
        {
            PS * LastPS=NULL;
            PS * SecLastPS=NULL;
            LastPS = m_FirstActualPS;
            // get to end of list
            while (LastPS)
            {
                SecLastPS = LastPS;
                LastPS = LastPS -> NextPS;
            }
            //SecLastPS is the last PS
            SecLastPS->NextPS = NewPS;
        }//end else
        TempPS = TempPS -> NextPS;
    }//end while
}//end function StoreSources

void CIsocurveDoc::OnEditStoreEnergy()
{
    if (m_StoredArea)
    {
        delete [] m_StoredArea;
        m_StoredArea = NULL;
    }

    if (m_LoadedFile)
    {
        CalculateDose();
        m_StoredArea = new DoseArea[m_xSize*m_ySize];
        for (int i=0;i<m_xSize;i++)
        {
            for (int j=0;j<m_ySize;j++)
            {
                CString s;
                s.Format("%f",m_Area[i*m_ySize + j].Dose);
                //AfxMessageBox(s);
                m_StoredArea[i*m_ySize + j].Dose = m_Area[i*m_ySize + j].Dose;
                s.Format("%f",m_StoredArea[i*m_ySize + j].Dose);
                //AfxMessageBox(s);
                m_StoredArea[i*m_ySize + j].Type = m_Area[i*m_ySize + j].Type;
            }//end for
        }//end for
    }
    else
        AfxMessageBox("There is none loaded, please load a file first");
}//end function StoreEnergy

void CIsocurveDoc::OnViewMultfact()
{

    CMultFactorDlg dlg;

    dlg.m_MultFact = m_MultFact;
    if (dlg.DoModal() == IDOK)
    {
        m_MultFact = dlg.m_MultFact;
    }
}

void CIsocurveDoc::OnEditClearPS()
{

    //for each source delete
    int NumSources = NumPS(m_FirstPS);

    CString s;
    s.Format("%d",NumSources);
```

```
    AfxMessageBox(s);
    for (int i = 0;i<NumSources;i++)
        DeletePointSource(m_FirstPS->x,m_FirstPS->y);


}

void CIsocurveDoc::OnPrescribe()
{
    OnViewIsocurve();
}

void CIsocurveDoc::OnShowhideiso()
{
    OnViewToggleisocurve();
}

void CIsocurveDoc::OnViewToggleisocurve()
{
    if (!m_IsocurveSet)
    {
        if (!m_LoadedFile)
            AfxMessageBox("Please load a file first.");
        else if (m_FirstPS == NULL)
            AfxMessageBox("There are no point sources loaded, please create a point source, then plot an
isocurve.");
        else
        {
            m_IsocurveSet = TRUE;
            m_NewIso = TRUE;
            CalculateDose();
        }
    }
    else
        m_IsocurveSet = FALSE;
    UpdateAllViews(NULL);
}

void CIsocurveDoc::OnFileSaveAs()
{
    CFileDialog Dlg(FALSE,"*.pgm","*.pgm",OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,"PGM Files
(*.pgm)|*.pgm||");

    if (Dlg.DoModal() == IDOK)
    {
        m_FileName = Dlg.GetPathName();
        // now write out the file...
    }
    else
        return;

    FILE * OutputFile=NULL;
    OutputFile = fopen(m_FileName,"w");

    fprintf(OutputFile,"P2\n");/* P2 */
    fprintf(OutputFile,"# CREATOR: Steve Miller, 1999\n");/* # CREATOR: XV Version 3.01  Rev: 3/30/93 */
    /* now put the x and y dimensions of the file */
    fprintf(OutputFile,"%d %d\n",m_xSize,m_ySize);
    // number of colors
    fprintf(OutputFile,"255\n");/* 255 */

    int counter=0;
    for (int i=0;i<m_xSize;i++)
    {
        for (int j=0;j<m_ySize;j++)
        {
            fprintf(OutputFile,"%d ",m_Area[(i*m_ySize)+j].Type);
            counter++;
            if (counter==17)
            {
                fprintf(OutputFile,"\n");
                counter=0;
```

```
            }
        }
    }

    fclose(OutputFile);
}

void CIsocurveDoc::OnFileOpen()
{
    OnFileNew();
}
```

E.25 IsocurveDoc.h

```
// IsocurveDoc.h : interface of the CIsocurveDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_ISOCURVEDOC_H__E8F0A3AC_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_ISOCURVEDOC_H__E8F0A3AC_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_

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


class CIsocurveDoc : public COleDocument
{
protected: // create from serialization only
    CIsocurveDoc();
    DECLARE_DYNCREATE(CIsocurveDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CIsocurveDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    float m_MultFact;
    BOOL m_StoredDesired;
    void StoreSources();
    void SeedRandom();
    void MovePS(int WhichSource,int Distance);
    void MoveSourcesAround(int Distance);
    void MakeSquare(int x,int y);
    void MakeCircle(int radius);
    void GenerateStats();
    float m_MaxVSMinIndex;
    float m_HyperIndex;
    float m_HomoIndex;
    float m_PredMax;
    BOOL m_Coons;
    BOOL m_NewIso;
    void DeleteAllPredictedPoints();
    void AddPredictedPoint(int x,int y,float DwellTime);
    BOOL m_ViewPredictions;
    float m_Ratio;
```

```
    int m_MinOutsideY;
    int m_MinOutsideX;
    int m_MaxY;
    int m_MaxX;
    void AddDesiredPointSource(int x,int y,float DwellTime);
    int m_TumourCount;
    void DeletePointSource(int i,int j);
    void AddPointSource(int x,int y,float DwellTime);
    int m_LoadedFile;
    CString m_FileName;
    int m_MinY;
    int m_MinX;
    float Distance(int x1,int y1,int x2,int y2);
    BOOL m_IsocurveSet;
    float m_Isovalue;
    int m_ySize;
    int m_xSize;
    virtual ~CIsocurveDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    typedef struct _DoseArea{
        float Dose;
        float Pred;
        int Type;
    }DoseArea;

    typedef struct _PointSource{
        int x;
        int y;
        float DwellTime;
        _PointSource *NextPS;
    }PS;

    PS * m_FirstPS;
    PS * m_FirstActualPS;
    PS * m_TempPS;//this is so that the view can go through the PS easily
    PS * m_FirstPredictedPS;
    PS * m_FirstDesiredPS;
    void CalculateDose();
    DoseArea *m_Area;
    DoseArea *m_StoredArea;
    int NumPS(PS * ThePSList);
protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CIsocurveDoc)
    afx_msg void OnFileNew();
    afx_msg void OnViewIsocurve();
    afx_msg void OnViewClearisocurve();
    afx_msg void OnFileOutput();
    afx_msg void OnFileInputpredicted();
    afx_msg void OnViewRatio();
    afx_msg void OnViewNeuralNetwork();
    afx_msg void OnViewViewpredictions();
    afx_msg void OnViewCoonslinear();
    afx_msg void OnViewStatsreport();
    afx_msg void OnFileNewshape();
    afx_msg void OnFileSave();
    afx_msg void OnEditAddpointsource();
    afx_msg void OnEditRestoresourcepositions();
    afx_msg void OnEditStoreEnergy();
    afx_msg void OnViewMultfact();
    afx_msg void OnEditClearPS();
    afx_msg void OnPrescribe();
    afx_msg void OnShowhideiso();
    afx_msg void OnViewToggleisocurve();
    afx_msg void OnFileSaveAs();
    afx_msg void OnFileOpen();
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_ISOCURVEDOC_H__E8F0A3AC_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.26 IsocurveView.cpp

```
// IsocurveView.cpp : implementation of the CIsocurveView class
//

#include "stdafx.h"
#include <math.h>
#include "Isocurve.h"

#include "IsocurveDoc.h"
#include "CntrItem.h"
#include "IsocurveView.h"
#include "TransformsDlg.h"
#include "CtrlNet.h"

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


// global variables
CCtrlNet * FirstCtrlPt=NULL;
CCtrlNet * FirstCoonsPt=NULL;
CCtrlNet * du=NULL;
CCtrlNet * dv=NULL;
CCtrlNet * tw=NULL;
CCtrlNet * FirstLinNorm=NULL;



const char * const CIsocurveView::_ErrorStrings[] =
{
    {"No Error"},// 0
    {"Unable to get a DC"},// 1
    {"ChoosePixelFormat failed"},// 2
    {"SelectPixelFormat failed"},// 3
    {"wglCreateContext failed"},// 4
    {"wglMakeCurrent failed"},// 5
    {"wglDeleteContext failed"},// 6
    {"Swap Bufferes failed"},// 7
};

/////////////////////////////////////////////////////////////////////////////
// CIsocurveView

IMPLEMENT_DYNCREATE(CIsocurveView, CView)

BEGIN_MESSAGE_MAP(CIsocurveView, CView)
    //{{AFX_MSG_MAP(CIsocurveView)
    ON_WM_DESTROY()
    ON_WM_SETFOCUS()
    ON_WM_SIZE()
    ON_COMMAND(ID_OLE_INSERT_NEW, OnInsertObject)
    ON_COMMAND(ID_CANCEL_EDIT_CNTR, OnCancelEditCntr)
    ON_WM_CREATE()
```

```
    ON_WM_ERASEBKGND()
    ON_WM_KEYDOWN()
    ON_COMMAND(ID_VIEW_TRANSFORMS, OnViewTransforms)
    ON_COMMAND(ID_VIEW_PERSPECTIVEORTHO, OnViewPerspectiveortho)
    ON_COMMAND(ID_ORTHOPERSPECT, OnOrthoperspect)
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIsocurveView construction/destruction

CIsocurveView::CIsocurveView():
    m_hRC(0), m_pDC(0), m_ErrorString(_ErrorStrings[0])
{
    m_pSelection = NULL;
    m_Animate = 0;
    // load the modalless dialog box
    m_pTransDlg = new CTransformsDlg;
    m_pTransDlg->Create(IDD_TRANSFORMS,this);
    m_Perspective = 0;
    FirstCtrlPt = NULL;
}

CIsocurveView::~CIsocurveView()
{
    m_pTransDlg->DestroyWindow();
    delete m_pTransDlg;

    // get rid of control net and coons net (ll)
    /*if (FirstCtrlPt)
    {
        CCtrlNet *pCurPt = FirstCtrlPt;

        while (FirstCtrlPt)
        {
            pCurPt = FirstCtrlPt->m_pNextCtrlPt;
            delete FirstCtrlPt;
            FirstCtrlPt = pCurPt;
        }
        FirstCtrlPt=NULL;
    }*/

    if (FirstCtrlPt)
        delete [] FirstCtrlPt;

    if (FirstCoonsPt)
        delete [] FirstCoonsPt;

    if (du)
        delete [] du;

    if (dv)
        delete [] dv;

    if (tw)
        delete [] tw;

    if (FirstLinNorm)
        delete [] FirstLinNorm;
}

BOOL CIsocurveView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs
    cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;
    return CView::PreCreateWindow(cs);
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CIsocurveView drawing

void CIsocurveView::OnDraw(CDC* pDC)
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    // TODO: also draw all OLE items in the document

    // Draw the selection at an arbitrary position.  This code should be
    //  removed once your real drawing code is implemented.  This position
    //  corresponds exactly to the rectangle returned by CIsocurveCntrItem,
    //  to give the effect of in-place editing.

    // TODO: remove this code when final draw code is complete.

    /*if (m_pSelection == NULL)
    {
        POSITION pos = pDoc->GetStartPosition();
        m_pSelection = (CIsocurveCntrItem*)pDoc->GetNextClientItem(pos);
    }
    if (m_pSelection != NULL)
        m_pSelection->Draw(pDC, CRect(10, 10, 210, 210));*/

    ::glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    PreRenderScene();

    ::glPushMatrix();
    RenderStockScene();
    //::glPopMatrix();

    //::glPushMatrix();
    RenderScene();
    ::glPopMatrix();

    ::glFinish();

    if (FALSE == ::SwapBuffers(m_pDC->GetSafeHdc() ) )
        SetError(7);
}

void CIsocurveView::OnInitialUpdate()
{
    CView::OnInitialUpdate();


    // TODO: remove this code when final selection model code is written
    m_pSelection = NULL;    // initialize selection

}

/////////////////////////////////////////////////////////////////////////////
// CIsocurveView printing

BOOL CIsocurveView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CIsocurveView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CIsocurveView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}
```

```
void CIsocurveView::OnDestroy()
{
    // Deactivate the item on destruction; this is important
    // when a splitter view is being used.
   CView::OnDestroy();
   COleClientItem* pActiveItem = GetDocument()->GetInPlaceActiveItem(this);
   if (pActiveItem != NULL && pActiveItem->GetActiveView() == this)
   {
      pActiveItem->Deactivate();
      ASSERT(GetDocument()->GetInPlaceActiveItem(this) == NULL);
   }

   if (FALSE == ::wglMakeCurrent(0,0))
       SetError(2);

   if (FALSE == ::wglDeleteContext(m_hRC))
       SetError(6);

   if (m_pDC)
       delete m_pDC;
}


/////////////////////////////////////////////////////////////////////////////
// OLE Client support and commands

BOOL CIsocurveView::IsSelected(const CObject* pDocItem) const
{
    // The implementation below is adequate if your selection consists of
    //  only CIsocurveCntrItem objects.  To handle different selection
    //  mechanisms, the implementation here should be replaced.

    // TODO: implement this function that tests for a selected OLE client item

    return pDocItem == m_pSelection;
}

void CIsocurveView::OnInsertObject()
{
    // Invoke the standard Insert Object dialog box to obtain information
    //  for new CIsocurveCntrItem object.
    COleInsertDialog dlg;
    if (dlg.DoModal() != IDOK)
        return;

    BeginWaitCursor();

    CIsocurveCntrItem* pItem = NULL;
    TRY
    {
        // Create new item connected to this document.
        CIsocurveDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        pItem = new CIsocurveCntrItem(pDoc);
        ASSERT_VALID(pItem);

        // Initialize the item from the dialog data.
        if (!dlg.CreateItem(pItem))
            AfxThrowMemoryException();  // any exception will do
        ASSERT_VALID(pItem);

        if (dlg.GetSelectionType() == COleInsertDialog::createNewItem)
            pItem->DoVerb(OLEIVERB_SHOW, this);

        ASSERT_VALID(pItem);

        // As an arbitrary user interface design, this sets the selection
        //  to the last item inserted.

        // TODO: reimplement selection as appropriate for your application

        m_pSelection = pItem;   // set selection to last inserted item
```

```
        pDoc->UpdateAllViews(NULL);
    }
    CATCH(CException, e)
    {
        if (pItem != NULL)
        {
            ASSERT_VALID(pItem);
            pItem->Delete();
        }
        AfxMessageBox(IDP_FAILED_TO_CREATE);
    }
    END_CATCH

    EndWaitCursor();
}

// The following command handler provides the standard keyboard
//  user interface to cancel an in-place editing session.  Here,
//  the container (not the server) causes the deactivation.
void CIsocurveView::OnCancelEditCntr()
{
    // Close any in-place active item on this view.
    COleClientItem* pActiveItem = GetDocument()->GetInPlaceActiveItem(this);
    if (pActiveItem != NULL)
    {
        pActiveItem->Close();
    }
    ASSERT(GetDocument()->GetInPlaceActiveItem(this) == NULL);
}

// Special handling of OnSetFocus and OnSize are required for a container
//  when an object is being edited in-place.
void CIsocurveView::OnSetFocus(CWnd* pOldWnd)
{
    COleClientItem* pActiveItem = GetDocument()->GetInPlaceActiveItem(this);
    if (pActiveItem != NULL &&
        pActiveItem->GetItemState() == COleClientItem::activeUIState)
    {
        // need to set focus to this item if it is in the same view
        CWnd* pWnd = pActiveItem->GetInPlaceWindow();
        if (pWnd != NULL)
        {
            pWnd->SetFocus();   // don't call the base class
            return;
        }
    }

    CView::OnSetFocus(pOldWnd);
}

void CIsocurveView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);
    COleClientItem* pActiveItem = GetDocument()->GetInPlaceActiveItem(this);
    if (pActiveItem != NULL)
        pActiveItem->SetItemRects();


    if (0>=cx || 0 >= cy)
        return;

    m_cx = cx;
    m_cy = cy;
    SetupViewport(cx,cy);


    // select the projection matrix and clear it
    ::glMatrixMode(GL_PROJECTION);
    ::glLoadIdentity();

    //select the viewing volume
    SetupViewingFrustrum(cx,cy);
```

```
    //switch back to the Modelview matrix and clear it
    ::glMatrixMode(GL_MODELVIEW);
    ::glLoadIdentity();

    //now perform any viewing transformations
    SetupViewingTransform();
}

/////////////////////////////////////////////////////////////////////////////
// CIsocurveView diagnostics

#ifdef _DEBUG
void CIsocurveView::AssertValid() const
{
    CView::AssertValid();
}

void CIsocurveView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CIsocurveDoc* CIsocurveView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CIsocurveDoc)));
    return (CIsocurveDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CIsocurveView message handlers

int CIsocurveView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == -1)
        return -1;

    // TODO: Add your specialized creation code here

    InitializeOpenGL();

    return 0;
}

BOOL CIsocurveView::OnEraseBkgnd(CDC* pDC)
{
    // TODO: Add your message handler code here and/or call default

    // this is the original
    //return CView::OnEraseBkgnd(pDC);

    return TRUE;// this will tell windows not to erase the background
}

void CIsocurveView::SetError( int e)
{
    // if there was no previous error,
    // then set this one
    if (_ErrorStrings[0] == m_ErrorString )
        m_ErrorString = _ErrorStrings[e];
}// end function SetError


BOOL CIsocurveView::InitializeOpenGL()
{
    m_pDC = new CClientDC(this);

    if (NULL == m_pDC)
    {
        SetError(1);
        return FALSE;
```

```
    }

    if (!SetupPixelFormat())
        return FALSE;

    if (0 == (m_hRC = ::wglCreateContext( m_pDC->GetSafeHdc() ) ) )
    {
        SetError(4);
        return FALSE;
    }

    if (FALSE == ::wglMakeCurrent(m_pDC->GetSafeHdc(),m_hRC) )
    {
        SetError(5);
        return FALSE;
    }

    // specify black as clear color
    ::glClearColor(0.0f,0.0f,0.0f,0.0f);
    //specify back of the buffer as clear depth
    ::glClearDepth(1.0f);
    // enable depth testing
    ::glEnable( GL_DEPTH_TEST );
    // turn on some lighting...
    ::glEnable(GL_LIGHTING);
    GLfloat ambient[] = {0.0f,0.0f,0.0f,1.0f};
    GLfloat diffuse[] = {1.0f,1.0f,1.0f,1.0f};
    GLfloat specular[] = {0.0f,0.0f,0.0f,1.0f};
    GLfloat position[] = {-2.0f,-2.0f,-2.0f,0.0f};
    ::glEnable(GL_LIGHT0);
    ::glLightfv(GL_LIGHT0,GL_AMBIENT,ambient);
    ::glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuse);
    ::glLightfv(GL_LIGHT0,GL_SPECULAR,specular);
    ::glLightfv(GL_LIGHT0,GL_POSITION,position);

    //setup material properties.
    GLfloat materialSpecular[4] = {0.3f,1.0f,0.3f,1.0f};
    GLfloat materialShininess[1] = {128.0f};
    GLfloat materialAmbient[4] = {0.125f,0.5f,0.125f,1.0f};
    GLfloat materialDiffuse[4] = {0.2f,0.4f,0.2f,1.0f};

    return TRUE;
}


BOOL CIsocurveView::SetupPixelFormat()
{

    PIXELFORMATDESCRIPTOR pfd =
        {
            sizeof(PIXELFORMATDESCRIPTOR),//size of the pfd
            1,
            PFD_DRAW_TO_WINDOW |// view support
            PFD_SUPPORT_OPENGL |// support OPENGL
            PFD_DOUBLEBUFFER,// double buffered
            PFD_TYPE_RGBA,
            24,
            0,0,0,0,0,0,
            0,
            0,
            0,
            0,0,0,0,
            16,
            0,
            0,
            PFD_MAIN_PLANE,
            0,
            0,0,0
        };
    int pixelformat;

    if (0 == (pixelformat = ::ChoosePixelFormat(m_pDC->GetSafeHdc(), &pfd)))
```

```
    {
        SetError(2);
        return FALSE;
    }

    if (FALSE == ::SetPixelFormat(m_pDC->GetSafeHdc(),pixelformat,&pfd))
    {
        SetError(3);
        return FALSE;
    }

    return TRUE;
}// end function SetupPixelFormat()


BOOL CIsocurveView::SetupViewport(int cx,int cy)
{
    //select the client area
    glViewport (0,0,cx,cy);
    return TRUE;
}//end function SetupViewport

BOOL CIsocurveView::SetupViewingFrustrum(int cx,int cy)
{
    //select default viewing volume
    if (m_Perspective)
    {
        gluPerspective(45.0f,(cx/cy),0.1f,2000.0f);
        ::glEnable(GL_LIGHTING);
    }
    else
    {
        //glOrtho(-5,5,5,-5,-10.0f,50.0f);//left,right,bottom,top,near,far
        glOrtho(-5,5,5,-5,50.0f,-10.0f);//left,right,bottom,top,near,far
        ::glDisable(GL_LIGHTING);
    }
    return TRUE;
}//end function SetupViewingFrustrum

BOOL CIsocurveView::SetupViewingTransform()
{
    // select default viewing transformation
    // of a 20 degree rotation about the x axis
    // then a -5 unit transformation along the z axis
    if (m_Perspective)
    {
        ::glTranslatef(0.0f,0.0f,-20.0f);
    //::glRotatef(70.0f,1.0f,0.0f,0.0f);
        ::glRotatef(45.0f,1.0f,0.0f,0.0f);
    }
    else
    {
        //::glTranslatef(0.0f,0.0f,-2.0f);
        //::glRotatef(180.0f,1.0f,0.0f,0.0f);
    }
    //AfxMessageBox("SetingUpViewingTransform");
    return TRUE;
}//end function SetupViewingTransform


BOOL CIsocurveView::RenderScene()
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    float xPixelSize=0.0f,yPixelSize=0.0f;
    xPixelSize = 10.0f/(float)(pDoc->m_xSize);
    yPixelSize = 10.0f/(float)(pDoc->m_ySize);

    int value;
    GLfloat color[3];
    GLfloat v0[3],v1[3],v2[3],v3[3];
```

```
    GLfloat n0[3],n1[3],n2[3],n3[3];


    // this is the linear surface stuff
    if (pDoc->m_Area && pDoc->m_IsocurveSet && !pDoc->m_Coons)
    {
        float energy=0.0f;
        if (pDoc->m_NewIso)
        {
            pDoc->m_NewIso=FALSE;
            CreateControlNet();// call the control net calculation routine
            CreateNormArray();// call the normal calculation routine
        }

        for (int i=0;i<pDoc->m_xSize-1;i++)
        {
            for (int j=0;j<pDoc->m_ySize-1;j++)
            {
                glBegin( GL_QUADS );
                if ((i==pDoc->m_MinX) && (j==pDoc->m_MinY))// set the color to red for minimum on tumour
                    ::glColor3f(1.0f,0.0f,0.0f);
                else if (pDoc->m_Area[i*pDoc->m_ySize+j].Dose>=pDoc->m_Isovalue)// set the color to
green for stuff greater than isovalue
                    ::glColor3f(0.0f,1.0f,0.0f);
                else
                {
                    value = pDoc->m_Area[i*pDoc->m_ySize+j].Type;
                    color[0] = (float)value/(float)256;
                    color[1] = (float)value/(float)256;
                    color[2] = (float)value/(float)256;
                    ::glColor3fv(color);
                }

                if (!FirstCtrlPt)
                    return FALSE;
                if (!FirstLinNorm)
                    return FALSE;

                v0[0] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[0];
                v0[1] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[1];
                v0[2] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[2];
                v1[0] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[0];
                v1[1] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[1];
                v1[2] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[2];
                v2[0] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
                v2[1] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
                v2[2] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
                v3[0] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[0];
                v3[1] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[1];
                v3[2] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[2];

                n0[0] = FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[0];
                n0[1] = FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[1];
                n0[2] = FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[2];
                n1[0] = FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[0];
                n1[1] = FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[1];
                n1[2] = FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[2];
                n2[0] = FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
                n2[1] = FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
                n2[2] = FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
                n3[0] = FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[0];
                n3[1] = FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[1];
                n3[2] = FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[2];

                ::glNormal3fv( n0 );
                ::glVertex3fv( v0 );
                ::glNormal3fv( n1 );
                ::glVertex3fv( v1 );
                ::glNormal3fv( n2 );
                ::glVertex3fv( v2 );
                ::glNormal3fv( n3 );
                ::glVertex3fv( v3 );
```

```
                glEnd();
            }//end for i
        }//end for j
    }//end if

    //this is the Coons surface stuff
    else if (pDoc->m_Area && pDoc->m_IsocurveSet && pDoc->m_Coons)// if Coons
    {
        int n=0,m=0;
        if (pDoc->m_NewIso)
        {
            pDoc->m_NewIso=FALSE;
            CreateControlNet();// call the control net calculation routine
            CreateCoonsNet(n,m);// call the routine to subdivide the control net into coons patches
        }
        for (int i=0;i<pDoc->m_xSize-1;i++)
        {
            for (int j=0;j<pDoc->m_ySize-1;j++)
            {

                for (int q=0;q<10;q++)
                {
                    for (int r=0;r<10;r++)
                    {
                        if ((i==pDoc->m_MinX) && (j==pDoc->m_MinY))// set the color to red for minimum
on tumour
                        {
                            glBegin( GL_QUADS );
                            ::glColor3f(1.0f,0.0f,0.0f);

                        }
                        else if (pDoc->m_Area[i*pDoc->m_ySize+j].Dose>=pDoc->m_Isovalue)// set the color
to green for stuff greater than isovalue
                        {
                            glBegin( GL_QUADS );
                            ::glColor3f(0.0f,1.0f,0.0f);
                        }
                        else
                        {
                            value = pDoc->m_Area[i*pDoc->m_ySize+j].Type;
                            glBegin( GL_QUADS );
                            color[0] = (float)value/(float)256;
                            color[1] = (float)value/(float)256;
                            color[2] = (float)value/(float)256;
                            ::glColor3fv(color);
                        }

                        int index = q+r*n*pDoc->m_xSize+j*pDoc->m_xSize*n*m+i*n;
                        v0[0] = FirstCoonsPt[index].m_Pos[0];
                        v0[1] = FirstCoonsPt[index].m_Pos[1];
                        v0[2] = FirstCoonsPt[index].m_Pos[2];
                        v1[0] = FirstCoonsPt[index+n].m_Pos[0];
                        v1[1] = FirstCoonsPt[index+n].m_Pos[1];
                        v1[2] = FirstCoonsPt[index+n].m_Pos[2];
                        v2[0] = FirstCoonsPt[index+m*n*pDoc->m_xSize].m_Pos[0];
                        v2[1] = FirstCoonsPt[index+m*n*pDoc->m_xSize].m_Pos[1];
                        v2[2] = FirstCoonsPt[index+m*n*pDoc->m_xSize].m_Pos[2];
                        v3[0] = FirstCoonsPt[index+m*n*pDoc->m_xSize+n].m_Pos[0];
                        v3[1] = FirstCoonsPt[index+m*n*pDoc->m_xSize+n].m_Pos[1];
                        v3[2] = FirstCoonsPt[index+m*n*pDoc->m_xSize+n].m_Pos[2];


                        ::glVertex3fv( v0 );
                        ::glVertex3fv( v1 );
                        ::glVertex3fv( v2 );
                        ::glVertex3fv( v3 );
                        glEnd();
                    }// end for r
                }// end for q
            }//end for j
        }//end for i
    }//end if Coons
```

```
    return TRUE;
}// end render scene

void CIsocurveView::RenderStockScene()
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    static float rotation = 0.0f;

    if (m_Animate)
    {
        rotation+=2.0;
        m_Animate = 0;
    }

    if (!m_Perspective)
        rotation = 0.0f;

    // setup dynamic materials..... (pg 200 fosner)
    ::glEnable(GL_COLOR_MATERIAL);
    ::glColorMaterial(GL_FRONT,GL_SHININESS);

    ::glRotatef(rotation,1.0f,0.0f,0.0f);


    float xPixelSize=0.0f,yPixelSize=0.0f;
    xPixelSize = 10.0f/(float)(pDoc->m_xSize);
    yPixelSize = 10.0f/(float)(pDoc->m_ySize);
    GLfloat color[3],v0[3],v1[3],v2[3],v3[3];

    for (int i=0;i<pDoc->m_xSize;i++)
    {
        for (int j=0;j<pDoc->m_ySize;j++)
        {
            int value = pDoc->m_Area[i*pDoc->m_ySize+j].Type;

            glBegin( GL_QUADS );
            color[0] = (float)value/(float)256;
            color[1] = (float)value/(float)256;
            color[2] = (float)value/(float)256;
            ::glColor3fv(color);

            v0[0] = -5.0f+xPixelSize*i;
            v0[1] = -5.0f+yPixelSize*j;
            v0[2] = 0.0f;

            v1[0] = v0[0]+xPixelSize;
            v1[1] = v0[1];
            v1[2] = 0.0f;

            v2[0] = v0[0]+xPixelSize;
            v2[1] = v0[1]+yPixelSize;
            v2[2] = 0.0f;

            v3[0] = v0[0];
            v3[1] = v0[1]+yPixelSize;
            v3[2] = 0.0f;

            ::glVertex3fv( v0 );
            ::glVertex3fv( v1 );
            ::glVertex3fv( v2 );
            ::glVertex3fv( v3 );
            glEnd();
        }
    }

    // add predicted points
    if (pDoc->m_ViewPredictions)
    {
        for (i=4;i<=pDoc->m_xSize-7;i++)
        {
```

```
            for (int j=4;j<=pDoc->m_ySize-7;j++)
            {
                //int value = pDoc->m_Area[i*pDoc->m_ySize+j].Type;

                glBegin( GL_QUADS );
                color[0] = 0.0;
                color[1] = 0.0f;
                //color[2] = 1.0f*pDoc->m_Area[(i)*pDoc->m_ySize+j].Pred;
                color[2] = 1.0f;
                ::glColor3fv(color);
                v0[0] = -5.0f+xPixelSize*i;
                v0[1] = -5.0f+yPixelSize*j;
                v0[2] = -pDoc->m_Area[(i)*pDoc->m_ySize+j].Pred;

                color[0] = 0.0;
                color[1] = 0.0f;
                //color[2] = 1.0f*pDoc->m_Area[(i+1)*pDoc->m_ySize+j].Pred;
                color[2] = 1.0f;
                ::glColor3fv(color);
                v1[0] = v0[0]+xPixelSize;
                v1[1] = v0[1];
                v1[2] = -pDoc->m_Area[(i+1)*pDoc->m_ySize+j].Pred;

                color[0] = 0.0;
                color[1] = 0.0f;
                //color[2] = 1.0f*pDoc->m_Area[(i+1)*pDoc->m_ySize+j+1].Pred;
                color[2] = 1.0f;
                ::glColor3fv(color);
                v2[0] = v0[0]+xPixelSize;
                v2[1] = v0[1]+yPixelSize;
                v2[2] = -pDoc->m_Area[(i+1)*pDoc->m_ySize+j+1].Pred;

                color[0] = 0.0;
                color[1] = 0.0f;
                //color[2] = 1.0f*pDoc->m_Area[(i)*pDoc->m_ySize+j+1].Pred;
                color[2] = 1.0f;
                ::glColor3fv(color);
                v3[0] = v0[0];
                v3[1] = v0[1]+yPixelSize;
                v3[2] = -pDoc->m_Area[(i)*pDoc->m_ySize+j+1].Pred;

                //give shading a shot
                GLfloat normal[3];
                CalcNormVec3D(v2,v1,v0,normal);
                ::glNormal3fv(normal);
                ::glVertex3fv( v0 );
                ::glNormal3fv(normal);
                ::glVertex3fv( v1 );
                ::glNormal3fv(normal);
                ::glVertex3fv( v2 );
                ::glNormal3fv(normal);
                ::glVertex3fv( v3 );
                glEnd();
            }// end for j
        }//end for i
    }// end if pred

    //::glDisable(GL_COLOR_MATERIAL);  //leave enabled to have colors.
}//end StockScene


int CIsocurveView::round(float x)
{
    return((int)ceil(x));
}


void CIsocurveView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    if (m_Animate)
        m_Animate = 0;
    else
```

```
        m_Animate = 1;
    Invalidate();
    CView::OnKeyDown(nChar, nRepCnt, nFlags);
}


void CIsocurveView::OnViewTransforms()
{
    m_pTransDlg->m_XT = 50;
    if (m_pTransDlg->IsWindowVisible())
        m_pTransDlg->SetFocus();
    else
        m_pTransDlg->ShowWindow(SW_SHOW);
    m_pTransDlg->UpdateWindow();
}

void CIsocurveView::OnViewPerspectiveortho()
{
    if (!m_Perspective)
        m_Perspective = 1;
    else
        m_Perspective = 0;
    OnSize(0,m_cx,m_cy);
    Invalidate();
}



void CIsocurveView::CreateControlNet()
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    int x=0,y=0;
    float xPixelSize=0.0f,yPixelSize=0.0f;

    // delete old control net
    if (FirstCtrlPt)
    {
        delete [] FirstCtrlPt;
        FirstCtrlPt = NULL;
    }

    FirstCtrlPt = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize];

    //calculate window pixel sizes
    xPixelSize = 10.0f/(float)(pDoc->m_xSize);
    yPixelSize = 10.0f/(float)(pDoc->m_ySize);
    float denom=0.f;
    denom = pDoc->m_Area[pDoc->m_MaxX*pDoc->m_ySize+pDoc->m_MaxY].Dose - pDoc->m_Area[pDoc-
>m_MinOutsideX*pDoc->m_ySize+pDoc->m_MinOutsideY].Dose;
    for (int i=0;i<pDoc->m_xSize;i++)
    {
        for (int j=0;j<pDoc->m_ySize;j++)
        {
            // calculate the normalized energy
            float energy;

            if (denom>0.01)
                energy = ((pDoc->m_Area[i*pDoc->m_ySize+j].Dose - pDoc->m_Area[pDoc-
>m_MinOutsideX*pDoc->m_ySize+pDoc->m_MinOutsideY].Dose)/denom)*10.0f;
            else
                energy = 0.0f;

            FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[0] = -5.0f+xPixelSize*i;
            FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[1] = -5.0f+yPixelSize*j;
            FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[2] = -(energy+0.3);
        }
    }

}
```

```
void CIsocurveView::CreateCoonsNet(int n, int m)
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);


    // Setup du dv tw
    if (du)
    {
        delete [] du;
        du = NULL;
    }

    if (dv)
    {
        delete [] dv;
        du = NULL;
    }

    if (tw)
    {
        delete [] tw;
        du = NULL;
    }
    if (FirstCoonsPt)
    {
        delete [] FirstCoonsPt;
        FirstCoonsPt = NULL;
    }

    du = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize];
    dv = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize];
    tw = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize];
    FirstCoonsPt = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize*n*m];

// du- notice special method of running through them (y then x)
    CCtrlNet to1,from1,to2,from2;
    for (int i=0;i<pDoc->m_ySize-1;i++)
    {
        CopyVec3D(du[i*pDoc->m_xSize],NormVec3D(SubtVec3D(AddVec3D(ScaleVec3D(FirstCtrlPt[i*pDoc-
>m_xSize],0.99),ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+1],0.01)),FirstCtrlPt[i*pDoc->m_xSize])));
        for (int j=1;j<pDoc->m_xSize-2;j++)
        {
            CopyVec3D(to1,FirstCtrlPt[i*pDoc->m_xSize+j]);
            CopyVec3D(from1,AddVec3D(ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+j-
1],0.01),ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+j],0.99)));
            CopyVec3D(to2,AddVec3D(ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+j+2],0.01),ScaleVec3D(FirstC-
trlPt[i*pDoc->m_xSize+j],0.99)));
            CopyVec3D(from2,FirstCtrlPt[i*pDoc->m_xSize+j]);
            CopyVec3D(du[i*pDoc-
>m_xSize+j],NormVec3D(ScaleVec3D(AddVec3D(SubtVec3D(to1,from1),SubtVec3D(to2,from2)),0.5)));
        }
        CopyVec3D(du[i*pDoc->m_xSize+pDoc->m_xSize-1],NormVec3D(SubtVec3D(FirstCtrlPt[i*pDoc-
>m_xSize+pDoc->m_xSize-1],AddVec3D(ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+pDoc->m_xSize-
1],0.99),ScaleVec3D(FirstCtrlPt[i*pDoc->m_xSize+pDoc->m_xSize-2],0.01)))));
    }

//dv
    for (int j=0;j<pDoc->m_xSize-1;j++)
    {
        CopyVec3D(dv[(pDoc->m_ySize-1)*pDoc->m_ySize+j],NormVec3D(SubtVec3D(AddVec3D(ScaleVec3D(FirstC-
trlPt[(pDoc->m_ySize-1)*pDoc->m_ySize+j],0.99),ScaleVec3D(FirstCtrlPt[(pDoc->m_ySize-2)*pDoc-
>m_ySize+j],0.01)),FirstCtrlPt[(pDoc->m_ySize-1)*pDoc->m_ySize+j])));
        for (int i=pDoc->m_ySize-2;i>0;i--)
        {
            CopyVec3D(to1,FirstCtrlPt[i*pDoc->m_ySize+j]);
            CopyVec3D(from1,AddVec3D(ScaleVec3D(FirstCtrlPt[(i+1)*pDoc-
>m_ySize+j],0.01),ScaleVec3D(FirstCtrlPt[i*pDoc->m_ySize+j],0.99)));
            CopyVec3D(to2,AddVec3D(ScaleVec3D(FirstCtrlPt[(i-1)*pDoc-
>m_ySize+j],0.01),ScaleVec3D(FirstCtrlPt[i*pDoc->m_ySize+j],0.99)));
            CopyVec3D(from2,FirstCtrlPt[i*pDoc->m_ySize+j]);
```

```
            CopyVec3D(dv[i*pDoc-
>m_ySize+j],NormVec3D(ScaleVec3D(AddVec3D(SubtVec3D(to1,from1),SubtVec3D(to2,from2)),0.5)));
        }
        CopyVec3D(dv[j],NormVec3D(SubtVec3D(FirstCtrlPt[j],AddVec3D(ScaleVec3D(FirstCtr-
lPt[j],0.99),ScaleVec3D(FirstCtrlPt[pDoc->m_ySize+j],0.01)))));
    }


// twist is soooo easy..
    for (i=0;i<pDoc->m_xSize-1;i++)
    {
        for (int j=0;j<pDoc->m_ySize-1;j++)
        {
            CopyVec3D(tw[j*pDoc->m_xSize+i],NormVec3D(CrossVec3D(du[j*pDoc->m_xSize+i],dv[j*pDoc-
>m_xSize+i])));
        }
    }

    GLfloat P00[3],P01[3],P10[3],P11[3];
    GLfloat dPdu00[3],dPdu01[3],dPdu10[3],dPdu11[3];
    GLfloat dPdv00[3],dPdv01[3],dPdv10[3],dPdv11[3];
    GLfloat tw00[3],tw01[3],tw10[3],tw11[3];

    // go through the control net exept the last x and y
    int xCount=0,yCount=0;
    for (i=0;i<pDoc->m_xSize-1;i++)
    {
        for (int j=0;j<pDoc->m_ySize-1;j++)
        {
            // setup vectors:
            P00[0] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[0];
            P00[1] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[1];
            P00[2] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i].m_Pos[2];
            P01[0] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[0];
            P01[1] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[1];
            P01[2] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[2];
            P10[0] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
            P10[1] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
            P10[2] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
            P11[0] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[0];
            P11[1] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[1];
            P11[2] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[2];

            dPdu00[0] = du[(j+1)*pDoc->m_xSize+i].m_Pos[0];
            dPdu00[1] = du[(j+1)*pDoc->m_xSize+i].m_Pos[1];
            dPdu00[2] = du[(j+1)*pDoc->m_xSize+i].m_Pos[2];
            dPdu01[0] = du[j*pDoc->m_xSize+i].m_Pos[0];
            dPdu01[1] = du[j*pDoc->m_xSize+i].m_Pos[1];
            dPdu01[2] = du[j*pDoc->m_xSize+i].m_Pos[2];
            dPdu10[0] = du[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
            dPdu10[1] = du[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
            dPdu10[2] = du[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
            dPdu11[0] = du[j*pDoc->m_xSize+i+1].m_Pos[0];
            dPdu11[1] = du[j*pDoc->m_xSize+i+1].m_Pos[1];
            dPdu11[2] = du[j*pDoc->m_xSize+i+1].m_Pos[2];

            dPdv00[0] = dv[(j+1)*pDoc->m_xSize+i].m_Pos[0];
            dPdv00[1] = dv[(j+1)*pDoc->m_xSize+i].m_Pos[1];
            dPdv00[2] = dv[(j+1)*pDoc->m_xSize+i].m_Pos[2];
            dPdv01[0] = dv[j*pDoc->m_xSize+i].m_Pos[0];
            dPdv01[1] = dv[j*pDoc->m_xSize+i].m_Pos[1];
            dPdv01[2] = dv[j*pDoc->m_xSize+i].m_Pos[2];
            dPdv10[0] = dv[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
            dPdv10[1] = dv[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
            dPdv10[2] = dv[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
            dPdv11[0] = dv[j*pDoc->m_xSize+i+1].m_Pos[0];
            dPdv11[1] = dv[j*pDoc->m_xSize+i+1].m_Pos[1];
            dPdv11[2] = dv[j*pDoc->m_xSize+i+1].m_Pos[2];

            tw00[0] = tw[(j+1)*pDoc->m_xSize+i].m_Pos[0];
            tw00[1] = tw[(j+1)*pDoc->m_xSize+i].m_Pos[1];
            tw00[2] = tw[(j+1)*pDoc->m_xSize+i].m_Pos[2];
```

```
            tw01[0] = tw[j*pDoc->m_xSize+i].m_Pos[0];
            tw01[1] = tw[j*pDoc->m_xSize+i].m_Pos[1];
            tw01[2] = tw[j*pDoc->m_xSize+i].m_Pos[2];
            tw10[0] = tw[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
            tw10[1] = tw[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
            tw10[2] = tw[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];
            tw11[0] = tw[j*pDoc->m_xSize+i+1].m_Pos[0];
            tw11[1] = tw[j*pDoc->m_xSize+i+1].m_Pos[1];
            tw11[2] = tw[j*pDoc->m_xSize+i+1].m_Pos[2];

            for (int q=0;q<=n;q++)
            {
                for (int r=0;r<=m;r++)
                {
                    double a,b,c,d,ex,fx,gx,hx,ey,fy,gy,hy,ez,fz,gz,hz,u,v;
                    u = (q/(double)n);
                    v = (r/(double)m);
                    a = 2.0*u*u*u-3.0*u*u+1.0;
                    b = -2.0*u*u*u+3.0*u*u;
                    c = u*u*u-2.0*u*u+u;
                    d = u*u*u-u*u;
                    ex = a*P00[0] + b*P10[0] + c*dPdu00[0] + d*dPdu10[0];
                    fx = a*P01[0] + b*P11[0] + c*dPdu01[0] + d*dPdu11[0];
                    gx = a*dPdv00[0] + b*dPdv10[0] + c*tw00[0] + d*tw10[0];
                    hx = a*dPdv01[0] + b*dPdv11[0] + c*tw01[0] + d*tw11[0];

                    ey = a*P00[1] + b*P10[1] + c*dPdu00[1] + d*dPdu10[1];
                    fy = a*P01[1] + b*P11[1] + c*dPdu01[1] + d*dPdu11[1];
                    gy = a*dPdv00[1] + b*dPdv10[1] + c*tw00[1] + d*tw10[1];
                    hy = a*dPdv01[1] + b*dPdv11[1] + c*tw01[1] + d*tw11[1];

                    ez = a*P00[2] + b*P10[2] + c*dPdu00[2] + d*dPdu10[2];
                    fz = a*P01[2] + b*P11[2] + c*dPdu01[2] + d*dPdu11[2];
                    gz = a*dPdv00[2] + b*dPdv10[2] + c*tw00[2] + d*tw10[2];
                    hz = a*dPdv01[2] + b*dPdv11[2] + c*tw01[2] + d*tw11[2];

                    CString s;
                    s.Format("%d",q+r*n*pDoc->m_xSize+j*pDoc->m_xSize*n*m+i*n);
                    //if (i>0)
                    //AfxMessageBox(s);
                    /*FirstCoonsPt[(j*m)+r+(i*n)+q].m_Pos[0] = (2.0*ex-2.0*fx+gx+hx)*v*v*v + (-
3.0*ex+3.0*fx-2.0*gx-hx)*v*v + gx*v + ex;
                    FirstCoonsPt[(j*m)+r+(i*n)+q].m_Pos[1] = (2.0*ey-2.0*fy+gy+hy)*v*v*v + (-
3.0*ey+3.0*fy-2.0*gy-hy)*v*v + gy*v + ey;
                    FirstCoonsPt[(j*m)+r+(i*n)+q].m_Pos[2] = (2.0*ez-2.0*fz+gz+hz)*v*v*v + (-
3.0*ez+3.0*fz-2.0*gz-hz)*v*v + gz*v + ez;*/
                    FirstCoonsPt[q+r*n*pDoc->m_xSize+j*pDoc->m_xSize*n*m+i*n].m_Pos[0] = (2.0*ex-
2.0*fx+gx+hx)*v*v*v + (-3.0*ex+3.0*fx-2.0*gx-hx)*v*v + gx*v + ex;
                    FirstCoonsPt[q+r*n*pDoc->m_xSize+j*pDoc->m_xSize*n*m+i*n].m_Pos[1] = (2.0*ey-
2.0*fy+gy+hy)*v*v*v + (-3.0*ey+3.0*fy-2.0*gy-hy)*v*v + gy*v + ey;
                    FirstCoonsPt[q+r*n*pDoc->m_xSize+j*pDoc->m_xSize*n*m+i*n].m_Pos[2] = (2.0*ez-
2.0*fz+gz+hz)*v*v*v + (-3.0*ez+3.0*fz-2.0*gz-hz)*v*v + gz*v + ez;
                    //AddSPoint(x,y,z,Ucount,Vcount);
                }
            }

        }// end for j
    }// end for i
}

CCtrlNet CIsocurveView::ScaleVec3D(CCtrlNet InVect, GLfloat Scale)
{
    InVect.m_Pos[0] *= Scale;
    InVect.m_Pos[1] *= Scale;
    InVect.m_Pos[2] *= Scale;
    return InVect;
}

CCtrlNet CIsocurveView::NormVec3D(CCtrlNet InVect)
{
    double denom;
```

```
    denom =
sqrt(InVect.m_Pos[0]*InVect.m_Pos[0]+InVect.m_Pos[1]*InVect.m_Pos[1]+InVect.m_Pos[2]*InVect.m_Pos[2]);
    InVect.m_Pos[0] = InVect.m_Pos[0]/denom;
    InVect.m_Pos[1] = InVect.m_Pos[1]/denom;
    InVect.m_Pos[2] = InVect.m_Pos[2]/denom;
    return InVect;

}

CCtrlNet CIsocurveView::SubtVec3D(CCtrlNet to, CCtrlNet from)
{
    CCtrlNet res;

    res.m_Pos[0] = to.m_Pos[0]-from.m_Pos[0];
    res.m_Pos[1] = to.m_Pos[1]-from.m_Pos[1];
    res.m_Pos[2] = to.m_Pos[2]-from.m_Pos[2];
    return res;
}

CCtrlNet CIsocurveView::AddVec3D(CCtrlNet a, CCtrlNet b)
{
    CCtrlNet res;

    res.m_Pos[0] = a.m_Pos[0]+b.m_Pos[0];
    res.m_Pos[1] = a.m_Pos[1]+b.m_Pos[1];
    res.m_Pos[2] = a.m_Pos[2]+b.m_Pos[2];
    return res;
}

CCtrlNet CIsocurveView::CrossVec3D(CCtrlNet a, CCtrlNet b)
{
    CCtrlNet t;

    t.m_Pos[0] = a.m_Pos[1]*b.m_Pos[2]-b.m_Pos[2]*a.m_Pos[2];
    t.m_Pos[1] = a.m_Pos[0]*b.m_Pos[2]-b.m_Pos[0]*a.m_Pos[2];
    t.m_Pos[2] = a.m_Pos[0]*b.m_Pos[1]-b.m_Pos[0]*a.m_Pos[1];
    return t;
}

void CIsocurveView::CopyVec3D(CCtrlNet a, CCtrlNet b)
{
    a.m_Pos[0] = b.m_Pos[0];
    a.m_Pos[1] = b.m_Pos[1];
    a.m_Pos[2] = b.m_Pos[2];
}

void CIsocurveView::CalcNormVec3D(GLfloat v0[3], GLfloat v1[3], GLfloat v2[3], GLfloat *(norm))
{
    GLfloat t1[3],t2[3],n[3],d;

    t1[0] = v1[0] - v0[0];
    t1[1] = v1[1] - v0[1];
    t1[2] = v1[2] - v0[2];
    t2[0] = v1[0] - v2[0];
    t2[1] = v1[1] - v2[1];
    t2[2] = v1[2] - v2[2];
    //cross products
    n[0] = t1[1]*t2[2] - t2[1]*t1[2];
    n[1] = t1[2]*t2[0] - t2[2]*t1[0];
    n[2] = t1[0]*t2[1] - t2[0]*t1[1];

    //normalize
    d = (n[0]*n[0] + n[1]*n[1] + n[2]*n[2]);

    // catch small ones
    if (d< 0.000001)
        d = 10000000.f;
    else
        d = 1.0f/sqrt(d);

    *norm = d*n[0];
    *(norm+1) = d*n[1];
```

```
    *(norm+2) = d*n[2];
}

void CIsocurveView::CreateNormArray()
{
    CIsocurveDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //create array
    if (FirstLinNorm)
    {
        delete [] FirstLinNorm;
        FirstLinNorm = NULL;
    }

    FirstLinNorm = new CCtrlNet[pDoc->m_xSize*pDoc->m_ySize];


    for (int i=0;i<pDoc->m_xSize-1;i++)
    {
        for (int j=0;j<pDoc->m_ySize-1;j++)
        {
            GLfloat v0[3],v1[3],v2[3];
            v0[0] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[0];
            v0[1] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[1];
            v0[2] = FirstCtrlPt[j*pDoc->m_xSize+i].m_Pos[2];
            v1[0] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[0];
            v1[1] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[1];
            v1[2] = FirstCtrlPt[j*pDoc->m_xSize+i+1].m_Pos[2];
            v2[0] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[0];
            v2[1] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[1];
            v2[2] = FirstCtrlPt[(j+1)*pDoc->m_xSize+i+1].m_Pos[2];

            GLfloat normal[3];
            CalcNormVec3D(v2,v1,v0,normal);
            FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[0] = normal[0];
            FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[1] = normal[1];
            FirstLinNorm[j*pDoc->m_xSize+i].m_Pos[2] = normal[2];
            FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[0] = normal[0];
            FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[1] = normal[1];
            FirstLinNorm[j*pDoc->m_xSize+i+1].m_Pos[2] = normal[2];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[0] = normal[0];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[1] = normal[1];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i+1].m_Pos[2] = normal[2];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[0] = normal[0];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[1] = normal[1];
            FirstLinNorm[(j+1)*pDoc->m_xSize+i].m_Pos[2] = normal[2];
        }//end for j
    }//end for i
}//end CreateNormArray

void CIsocurveView::OnOrthoperspect()
{
    OnViewPerspectiveortho();
}
```

E.27 IsocurveView.h

```
// IsocurveView.h : interface of the CIsocurveView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_ISOCURVEVIEW_H__E8F0A3AE_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_ISOCURVEVIEW_H__E8F0A3AE_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_

#include "CtrlNet.h"// Added by ClassView
#if _MSC_VER > 1000
#pragma once
```

```
#endif // _MSC_VER > 1000

#include "gl\gl.h"
#include "gl\glu.h"
#include "gl\glaux.h"
#include "TransformsDlg.h"// Added by ClassView
#include "CtrlNet.h"

class CIsocurveCntrItem;

class CIsocurveView : public CView
{
protected: // create from serialization only
    CIsocurveView();
    DECLARE_DYNCREATE(CIsocurveView)

// Attributes
public:
    CIsocurveDoc* GetDocument();
    // m_pSelection holds the selection to the current CIsocurveCntrItem.
    // For many applications, such a member variable isn't adequate to
    //  represent a selection, such as a multiple selection or a selection
    //  of objects that are not CIsocurveCntrItem objects.  This selection
    //  mechanism is provided just to help you get started.

    // TODO: replace this selection mechanism with one appropriate to your app.
    CIsocurveCntrItem* m_pSelection;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CIsocurveView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual BOOL IsSelected(const CObject* pDocItem) const;// Container support
    //}}AFX_VIRTUAL

// Implementation
public:
    void CreateNormArray();
    void CalcNormVec3D(GLfloat v0[3],GLfloat v1[3],GLfloat v2[3],GLfloat * (norm));
    void CopyVec3D(CCtrlNet a,CCtrlNet b);
    CCtrlNet CrossVec3D(CCtrlNet a,CCtrlNet b);
    CCtrlNet AddVec3D(CCtrlNet a,CCtrlNet b);
    CCtrlNet SubtVec3D(CCtrlNet to,CCtrlNet from);
    CCtrlNet NormVec3D(CCtrlNet InVect);
    CCtrlNet ScaleVec3D(CCtrlNet InVect,GLfloat Scale);
    void CreateCoonsNet(int n,int m);
    void CreateControlNet();
    int m_Perspective;
    CTransformsDlg * m_pTransDlg;
    int m_Animate;
    int round(float x);
    int m_cy;
    int m_cx;
    virtual ~CIsocurveView();
    virtual BOOL SetupPixelFormat( void );
    virtual BOOL SetupViewport( int cx, int cy );
    virtual BOOL SetupViewingFrustrum(  int cx, int cy  );
    virtual BOOL SetupViewingTransform( void );
    virtual BOOL RenderScene( void );
    virtual void RenderStockScene( void );
    virtual BOOL PreRenderScene( void ) { return TRUE; }
```

```
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

private:
    void SetError(int e);
    BOOL InitializeOpenGL();

    HGLRC m_hRC;
    CDC *m_pDC;

    static const char * const _ErrorStrings[];
    const char * m_ErrorString;

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CIsocurveView)
    afx_msg void OnDestroy();
    afx_msg void OnSetFocus(CWnd* pOldWnd);
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnInsertObject();
    afx_msg void OnCancelEditCntr();
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg BOOL OnEraseBkgnd(CDC* pDC);
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnViewTransforms();
    afx_msg void OnViewPerspectiveortho();
    afx_msg void OnOrthoperspect();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in IsocurveView.cpp
inline CIsocurveDoc* CIsocurveView::GetDocument()
   { return (CIsocurveDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_ISOCURVEVIEW_H__E8F0A3AE_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.28 Layer.cpp

```
// layer.cppV.Rao, H.Rao
// compile for floating point hardware if available
#include <stdio.h>
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "layer.h"


inline float squash(float input)
// squashing function
// use sigmoid -- can customize to something
// else if desired; can add a bias term too
//
{
if (input < -50)
    return 0.0;
elseif (input > 50)
```

```
        return 1.0;
    else return (float)(1/(1+exp(-(double)input)));

}


inline float randomweight(unsigned init)
{
int num;
// random number generator
// will return a floating point
// value between -1 and 1

if (init==1)// seed the generator
    srand ((unsigned)time(NULL));

num=rand() % 100;

return 2*(float(num/100.00))-1;
}

// the next function is needed for Turbo C++
// and Borland C++ to link in the appropriate
// functions for fscanf floating point formats:
static void force_fpf()
{
    float x, *y;
    y=&x;
    x=*y;
}

// ----------------------------------------
//             input layer
//----------------------------------------
input_layer::input_layer(int i, int o)
{



num_inputs=i;
num_outputs=o;

outputs = new float[num_outputs];
if (outputs==0)
    {
    cout << "not enough memory\n";
    cout << "choose a smaller architecture\n";
    exit(1);
    }
}

input_layer::~input_layer()
{
delete [num_outputs] outputs;
}


void input_layer::calc_out()
{
//nothing to do, yet
}



// ----------------------------------------
//             output layer
//----------------------------------------
```

```
output_layer::output_layer(int i, int o)
{

num_inputs=i;
num_outputs=o;
weights = new float[num_inputs*num_outputs];
output_errors = new float[num_outputs];
back_errors = new float[num_inputs];
outputs = new float[num_outputs];
expected_values = new float[num_outputs];
if ((weights==0)||(output_errors==0)||(back_errors==0)
    ||(outputs==0)||(expected_values==0))
    {
    cout << "not enough memory\n";
    cout << "choose a smaller architecture\n";
    exit(1);
    }

}

output_layer::~output_layer()
{
// some compilers may require the array
// size in the delete statement; those
// conforming to Ansi C++ will not
delete [num_outputs*num_inputs] weights;
delete [num_outputs] output_errors;
delete [num_inputs] back_errors;
delete [num_outputs] outputs;

}


void output_layer::calc_out()
{

int i,j,k;
float accumulator=0.0;


for (j=0; j<num_outputs; j++)
    {

    for (i=0; i<num_inputs; i++)

        {
        k=i*num_outputs;
        if (weights[k+j]*weights[k+j] > 1000000.0)
            {
            cout << "weights are blowing up\n";
            cout << "try a smaller learning constant\n";
            cout << "e.g. beta=0.02    aborting...\n";
            exit(1);
            }
        outputs[j]=weights[k+j]*(*(inputs+i));
        accumulator+=outputs[j];
        }
    // use the sigmoid squash function
    outputs[j]=squash(accumulator);
    accumulator=0;
    }

}


void output_layer::calc_error(float & error)
{
int i, j, k;
float accumulator=0;
float total_error=0;
```

```
for (j=0; j<num_outputs; j++)
    {
    output_errors[j] = expected_values[j]-outputs[j];
    total_error+=output_errors[j];
    }

error=total_error;

for (i=0; i<num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j<num_outputs; j++)
        {
        back_errors[i]=
            weights[k+j]*output_errors[j];
        accumulator+=back_errors[i];
        }
    back_errors[i]=accumulator;
    accumulator=0;
    // now multiply by derivative of
    // sigmoid squashing function, which is
    // just the input*(1-input)
    back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
    }

}

void output_layer::randomize_weights()
{
int i, j, k;
const unsigned first_time=1;

const unsigned not_first_time=0;
float discard;

discard=randomweight(first_time);

for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        weights[k+j]=randomweight(not_first_time);
    }
}

void output_layer::update_weights(const float beta)
{
int i, j, k;

// learning law: weight_change =
//      beta*output_error*input

for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        weights[k+j] +=
            beta*output_errors[j]*(*(inputs+i));
    }

}

void output_layer::list_weights()
{
int i, j, k;

for (i=0; i< num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        cout << "weight["<<i<<","<<
```

```
            j<<"] is: "<<weights[k+j];
    }

}

void output_layer::list_errors()
{
int i, j;

for (i=0; i< num_inputs; i++)
    cout << "backerror["<<i<<
        "] is : "<<back_errors[i]<<"\n";

for (j=0; j< num_outputs; j++)
    cout << "outputerrors["<<j<<
            "] is: "<<output_errors[j]<<"\n";

}


void output_layer::write_weights(int layer_no,
        FILE * weights_file_ptr)
{
int i, j, k;

// assume file is already open and ready for
// writing

// prepend the layer_no to all lines of data
// format:
//      layer_noweight[0,0] weight[0,1] ...
//      layer_noweight[1,0] weight[1,1] ...
//      ...

for (i=0; i< num_inputs; i++)
    {
    fprintf(weights_file_ptr,"%i ",layer_no);
    k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
    {
    fprintf(weights_file_ptr,"%f ",
            weights[k+j]);
    }
    fprintf(weights_file_ptr,"\n");
    }


}

void output_layer::read_weights(int layer_no,
        FILE * weights_file_ptr)
{
int i, j, k;


// assume file is already open and ready for
// reading

// look for the prepended layer_no
// format:
//      layer_noweight[0,0] weight[0,1] ...
//      layer_noweight[1,0] weight[1,1] ...
//      ...
while (1)

    {

    fscanf(weights_file_ptr,"%i",&j);
    if ((j==layer_no)|| (feof(weights_file_ptr)))
        break;
    else
        {
```

```
        while (fgetc(weights_file_ptr) != '\n')
            {;}// get rest of line
        }
    }

if (!(feof(weights_file_ptr)))
    {
    // continue getting first line
    i=0;
    for (j=0; j< num_outputs; j++)
            {

            fscanf(weights_file_ptr,"%f",
                    &weights[j]); // i*num_outputs = 0
        }
    fscanf(weights_file_ptr,"\n");



    // now get the other lines
    for (i=1; i< num_inputs; i++)
        {
        fscanf(weights_file_ptr,"%i",&layer_no);
        k=i*num_outputs;
    for (j=0; j< num_outputs; j++)
        {
        fscanf(weights_file_ptr,"%f",
            &weights[k+j]);
            }

    }
    fscanf(weights_file_ptr,"\n");
    }


else cout << "end of file reached\n";

}
void output_layer::list_outputs()
{
int j;

for (j=0; j< num_outputs; j++)
    {
    cout << "outputs["<<j
        <<"] is: "<<outputs[j]<<"\n";
    }

}


// ----------------------------------------
//             middle layer
//----------------------------------------


middle_layer::middle_layer(int i, int o):
    output_layer(i,o)
{

}

middle_layer::~middle_layer()
{
delete [num_outputs*num_inputs] weights;
delete [num_outputs] output_errors;
delete [num_inputs] back_errors;
delete [num_outputs] outputs;
}
```

```
void middle_layer::calc_error()
{
int i, j, k;
float accumulator=0;

for (i=0; i<num_inputs; i++)
    {
    k=i*num_outputs;
    for (j=0; j<num_outputs; j++)
        {
        back_errors[i]=
            weights[k+j]*(*(output_errors+j));
        accumulator+=back_errors[i];
        }
    back_errors[i]=accumulator;
    accumulator=0;
    // now multiply by derivative of
    // sigmoid squashing function, which is
    // just the input*(1-input)
    back_errors[i]*=(*(inputs+i))*(1-(*(inputs+i)));
    }

}



// ----------------------------------------
//             Network
//----------------------------------------

network::network()
{
position=0L;
}

network::~network()
{
int i,j,k;
i=layer_ptr[0]->num_outputs;// inputs
j=layer_ptr[number_of_layers-1]->num_outputs; //outputs
k=MAX_VECTORS;


delete [(i+j)*k]buffer;
}

void network::set_training(const unsigned & value)
{
training=value;
}

unsigned network::get_training_value()
{
return training;
}


void network::get_layer_info()
{
int i;

//----------------------------------------
//
//  Get layer sizes for the network
//
// ----------------------------------------



cout << " Please enter in the number of layers for your network.\n";
cout << " You can have a minimum of 3 to a maximum of 5. \n";
```

```
cout << " 3 implies 1 hidden layer; 5 implies 3 hidden layers : \n\n";

cin >> number_of_layers;

cout << " Enter in the layer sizes separated by spaces.\n";
cout << " For a network with 3 neurons in the input layer,\n";
cout << " 2 neurons in a hidden layer, and 4 neurons in the\n";
cout << " output layer, you would enter: 3 2 4 .\n";
cout << " You can have up to 3 hidden layers,for five maximum entries :\n\n";

for (i=0; i<number_of_layers; i++)
    {
    cin >> layer_size[i];
    }

// -------------------------------------------------------
// size of layers:
//      input_layer layer_size[0]
//      output_layerlayer_size[number_of_layers-1]
//      middle_layerslayer_size[1]
//                  optional: layer_size[number_of_layers-3]
//                  optional: layer_size[number_of_layers-2]
//--------------------------------------------------------


}

void network::set_layer_info(int numlay,int sizearr[10])
{
    number_of_layers = numlay;

    for (int i=0; i<number_of_layers; i++)
    {
        layer_size[i] = sizearr[i];
    }
}



void network::set_up_network()
{
int i,j,k;
//--------------------------------------------------------
// Construct the layers
//
//--------------------------------------------------------


layer_ptr[0] = new input_layer(0,layer_size[0]);

for (i=0;i<(number_of_layers-1);i++)
    {
    layer_ptr[i+1] =
    new middle_layer(layer_size[i],layer_size[i+1]);
    }

layer_ptr[number_of_layers-1] = new
output_layer(layer_size[number_of_layers-2],layer_size[number_of_layers-1]);

for (i=0;i<(number_of_layers-1);i++)
    {
    if (layer_ptr[i] == 0)
        {
        cout << "insufficient memory\n";
        cout << "use a smaller architecture\n";
        exit(1);
        }
    }

//--------------------------------------------------------
// Connect the layers
```

```
//
//-------------------------------------------------------
// set inputs to previous layer outputs for all layers,
//  except the input layer

for (i=1; i< number_of_layers; i++)
    layer_ptr[i]->inputs = layer_ptr[i-1]->outputs;



// for back_propagation, set output_errors to next layer
//      back_errors for all layers except the output
//      layer and input layer


for (i=1; i< number_of_layers -1; i++)
    ((output_layer *)layer_ptr[i])->output_errors =
        ((output_layer *)layer_ptr[i+1])->back_errors;

// define the IObuffer that caches data from
// the datafile
i=layer_ptr[0]->num_outputs;// inputs
j=layer_ptr[number_of_layers-1]->num_outputs; //outputs
k=MAX_VECTORS;

buffer=new
    float[(i+j)*k];
if (buffer==0)
    {
    cout << "insufficient memory for buffer\n";
    exit(1);
    }
}

void network::randomize_weights()
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->randomize_weights();
}


void network::update_weights(const float beta)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->update_weights(beta);
}


void network::write_weights(FILE * weights_file_ptr)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->write_weights(i,weights_file_ptr);
}


void network::read_weights(FILE * weights_file_ptr)
{
int i;

for (i=1; i<number_of_layers; i++)
    ((output_layer *)layer_ptr[i])
        ->read_weights(i,weights_file_ptr);
}
```

```
void network::list_weights()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_weights();
    }
}

void network::list_outputs()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_outputs();
    }
}

void network::write_outputs(FILE *outfile,FILE * m_OutputFile)
{
int i, ins, outs;
ins=layer_ptr[0]->num_outputs;
outs=layer_ptr[number_of_layers-1]->num_outputs;
float temp;

//MINE ADDED Sept. 9, 1999
int m_Count=0;
float MinOutputAtSourceLocation = 1.2f;
///////DONE MINE

/*fprintf(outfile,"for input vector:\n");

for (i=0; i<ins; i++)
    {
    temp=layer_ptr[0]->outputs[i];
    fprintf(outfile,"%.5f  ",temp);
    // MY ADDED CODE Sept. 9,1999
    if (temp == 1.0)// assume this is a source
    {
        m_Count++;
        if (layer_ptr[number_of_layers-1]->outputs[i]<MinOutputAtSourceLocation)
            MinOutputAtSourceLocation = layer_ptr[number_of_layers-1]->outputs[i];
    }
    ////////////// DONE MINE
    }


fprintf(outfile,"\noutput vector is (%d, %f):\n",m_Count,MinOutputAtSourceLocation);
*/
for (i=0; i<outs; i++)
    {
    temp=layer_ptr[number_of_layers-1]->
    outputs[i];
    fprintf(outfile,"%.5f  ",temp);

    }

/*/////MINE ADDED Sept. 9, 1999
fprintf(outfile,"\nMy output vector is:\n");

for (i=0; i<outs; i++)
    {
    temp=layer_ptr[number_of_layers-1]->
    outputs[i];
```

```
    if (temp>MinOutputAtSourceLocation)
        fprintf(outfile,"1.0 ");
    else
        fprintf(outfile,"0.0 ");
    }
int count=0;
for (i=0; i<outs; i++)
{
    temp=layer_ptr[number_of_layers-1]->outputs[i];

    if (count==30)
    {
        count=0;
        fprintf(m_OutputFile,"\n");
    }
    fprintf(m_OutputFile,"%f ",temp);
    count++;
}
    fprintf(m_OutputFile,"\n\n");
/////////DONE MINE*/



if (training==1)
{
fprintf(outfile,"\nexpected output vector is:\n");

for (i=0; i<outs; i++)
    {
    temp=((output_layer *)(layer_ptr[number_of_layers-1]))->
    expected_values[i];
    fprintf(outfile,"%f  ",temp);

    }
}

/*fprintf(outfile,"\n----------------------\n");*/

}



void network::list_errors()
{
int i;

for (i=1; i<number_of_layers; i++)
    {
    cout << "layer number : " <<i<< "\n";
    ((output_layer *)layer_ptr[i])
        ->list_errors();
    }
}



int network::fill_IObuffer(FILE * inputfile)
{
// this routine fills memory with
// an array of input, output vectors
// up to a maximum capacity of
// MAX_INPUT_VECTORS_IN_ARRAY
// the return value is the number of read
// vectors


int i, k, count, veclength;

int ins, outs;

ins=layer_ptr[0]->num_outputs;
```

```
outs=layer_ptr[number_of_layers-1]->num_outputs;

if (training==1)
    veclength=ins+outs;
else
    veclength=ins;

count=0;
while  ((count<MAX_VECTORS)&&
        (!feof(inputfile)))
    {
    k=count*(veclength);
    for (i=0; i<veclength; i++)
        {
        fscanf(inputfile,"%f",&buffer[k+i]);
        }
    fscanf(inputfile,"\n");
    count++;
    }

if (!(ferror(inputfile)))
    return count;
else return -1; // error condition

}


void network::set_up_pattern(int buffer_index)
{
// read one vector into the network
int i, k;
int ins, outs;

ins=layer_ptr[0]->num_outputs;
outs=layer_ptr[number_of_layers-1]->num_outputs;
if (training==1)
    k=buffer_index*(ins+outs);
else
    k=buffer_index*ins;

for (i=0; i<ins; i++)
    layer_ptr[0]->outputs[i]=buffer[k+i];

if (training==1)
{
    for (i=0; i<outs; i++)

        ((output_layer *)layer_ptr[number_of_layers-1])->
            expected_values[i]=buffer[k+i+ins];
}

}



void network::forward_prop()
{
int i;
for (i=0; i<number_of_layers; i++)
    {
    layer_ptr[i]->calc_out(); //polymorphic
                // function
    }
}

void network::backward_prop(float & toterror)
{
int i;

// error for the output layer
((output_layer*)layer_ptr[number_of_layers-1])->
```

```
            calc_error(toterror);

// error for the middle layer(s)
for (i=number_of_layers-2; i>0; i--)
    {
    ((middle_layer*)layer_ptr[i])->
            calc_error();

    }

}
```

E.29 Layer.h

```
// layer.hV.Rao, H. Rao
// header file for the layer class heirarchy and
// the network class

#define MAX_LAYERS5
#define MAX_VECTORS100


class network;

class layer
{

protected:

    int num_inputs;
    int num_outputs;
    float *outputs;// pointer to array of outputs
    float *inputs; // pointer to array of inputs, which
                   // are outputs of some other layer

    friend network;

public:

    virtual void calc_out()=0;
};



class input_layer: public layer
{

private:


public:

    input_layer(int, int);
    ~input_layer();
    virtual void calc_out();

};

class middle_layer;

class output_layer:public layer
{
protected:

    float * weights;
    float * output_errors; // array of errors at output
    float * back_errors; // array of errors back-propagated
    float * expected_values;// to inputs
```

```
    friend network;

public:


    output_layer(int, int);
    ~output_layer();
    virtual void calc_out();
    void calc_error(float &);
    void randomize_weights();
    void update_weights(const float);
    void list_weights();
    void write_weights(int, FILE *);
    void read_weights(int, FILE *);
    void list_errors();
    void list_outputs();
};



class middle_layer:public output_layer
{

private:

public:
    middle_layer(int, int);
    ~middle_layer();
    void calc_error();
};


class network

{

private:

layer *layer_ptr[MAX_LAYERS];
    int number_of_layers;
    int layer_size[MAX_LAYERS];
    float *buffer;
    fpos_t position;
    unsigned training;

public:
    network();
    ~network();
    void set_training(const unsigned &);
    unsigned get_training_value();
    void get_layer_info();
    void network::set_layer_info(int ,int [10]);
    void set_up_network();
    void randomize_weights();
    void update_weights(const float);
    void write_weights(FILE *);
    void read_weights(FILE *);
    void list_weights();
    void write_outputs(FILE *,FILE *);
    void list_outputs();
    void list_errors();
    void forward_prop();
    void backward_prop(float &);
    int fill_IObuffer(FILE *);
    void set_up_pattern(int);

};
```

E.30 Main.c

```
/***********************************************/
/*                     */
/* This is the third draft of the Simulated  */
/* Annealing algorithm. It has dynamic number of*/
/* sources and can have dwell times (using con-*/
/* servation of energy.  Most parameters are */
/* set at the command line. The purpose of this */
/* version is to change the cost function so */
/* that it depends on hyper dose sleaves*/
/* and the min in the tumour.*/
/*                     */
/***********************************************/
/*                     */
/*                     */
/*  CREATED: 12/07/1999 - present*/
/*  AUTHOR : STEVEN MILLER*/
/*  Programmed for M.Sc. Thesis*/
/*  Version: 3.0.0  */
/*                     */
/***********************************************/


#include <stdio.h>
#include <time.h>
#include <math.h>
#include <stdlib.h>
#include "globals.h"



/* these are the global variables */
DoseArea *Area=NULL;/* This is the 2D array of structures */
int xSize=0;/* This is read in from the tumour input file */
int ySize=0;/* This is read in from the tumour input file */
PS *FirstPS=NULL;/* This is the linked list created in InsertSources */
PS *FirstNewPS=NULL;/* This is the linked list created from FirstPS in MoveSources */
float DWELL_TIME = DEFAULT_DWELL_TIME;/* global var. for dwell_time */
float PRESCRIBED_DOSE = DEFAULT_PRESCRIBED_DOSE;/* the prescribed dose... */

float INTERNAL_WEIGHT = DEFAULT_INTERNAL_WEIGHT;/* used to weight the contribution from internal dose
*/
float EXTERNAL_WEIGHT = DEFAULT_EXTERNAL_WEIGHT;/* used to weight the contribution from external dose
*/
float EXTERNAL_FACTOR = DEFAULT_EXTERNAL_FACTOR;/* this is the percentage of prescribed dose that can be
outside the tumour*/

float TEMP_REDUCTION_FACTOR = DEFAULT_TEMP_REDUCTION_FACTOR;/* this is the factor by which to decrease
the temperature */
int MAX_ITERATIONS = DEFAULT_MAX_ITERATIONS;/* the number of iterations at each temperature */
int STOP_COUNT = DEFAULT_STOP_COUNT;/* if this many within WITHIN_FACTOR then stop */
float WITHIN_FACTOR = DEFAULT_WITHIN_FACTOR;/* this is the % of iterations that have to be within */
int GlobalTumourCount = 0;/* this is to count the number of locations that the sources can go */
int HyperDose = 0;/* This is the global variable to set if there is too much dose */

int TempGlobal = 0;

/*************************************************************************/
/*                 Function: Round*/
/*                                 */
/*  Purpose: To round floats into integers*/
/*  Input: Any float       */
/*  Output: An integer     */
/*                                 */
/*************************************************************************/
int round(float x)
{
return((int)ffloor(x+0.5));
}/*end function round*/
```

```
/*************************************************************************/
/*                  Function: distance*/
/*                                    */
/*  Purpose : To calculate the distance between 2 2D points*/
/*  Input : The two points (X1,Y1) and (X2,Y2) for which the dist.*/
/*  between them is required.*/
/*  Output : The distance as a float.*/
/*                                    */
/*************************************************************************/
float distance(int x1,int y1,int x2,int y2)
{
if ((x1-x2==0) && (y1-y2==0))
    return(0.0);
else
{
    return((float)sqrt(sqr(x1-x2)+sqr(y1-y2)));
}
}/*end of distance function*/



/*************************************************************************/
/*                  Function: GetInitialTemp*/
/*                                    */
/*  Purpose : This function will return the starting temperature*/
/*  for the specific case of simulated annealing in question.*/
/*  Inputs : None.              */
/*  Outputs : Float which is the initial temperature.*/
/*                                    */
/*************************************************************************/
float GetInitialTemp(int NumSources)
{
float Cost=0.0f;
float OldCost=0.0f;
float CurTemp=0.0f;
float DeltaE = 0.0f;
int M1=0;
int M2=0;
int i=0;
int PrettyOutput=0;

/* Run through a number of random seed placements and calculate the M1 M2 DeltaF values*/
/* From these values, calculate a value for InitialTemperature*/

#if DEBUG_INIT_TEMP
    fprintf(stderr,"GetInitTemp starting\n");
#endif

FreePS(&FirstPS);
/*Get an initial OldCost*/
InsertSources(NumSources);/* Insert N sources into the simulation */
OldCost = CalculateCost(FirstPS);/* Calculate Cost for Old cost */
FreePS(&FirstPS);

CurTemp = DEFAULT_TEMPERATURE;/* Set the default temperature */

M1=0;
M2=0;

#if DEBUG_INIT_TEMP
    fprintf(stderr,"In GetInitTemp, going into main loop\n");
#endif

fprintf(stderr,"Heating up temperature!\n");
while (1)
{
    for (i=0;i<MAX_INIT_TEMP_ITERATIONS;i++)
    {
        InsertSources(NumSources);/*Insert N sources into the simulation*/
        Cost = CalculateCost(FirstPS);/*Calculate the cost*/
```

```
        DeltaE = Cost - OldCost;/*Find the difference*/

        if (DeltaE <= 0.0f)/*If DeltaE is negative, new cost is better so keep it*/
        {
            OldCost = Cost;
            M1++;
        }/*end if*/
        else if (((float)rand()/RAND_MAX) < (exp((-DeltaE)/CurTemp)))/*else keep it with some prob.*/
        {
            OldCost = Cost;
            M2++;
        }/*end else if*/

        FreePS(&FirstPS); /*remove the point sources */
    }/*end for loop*/

    if (((float)(M1+M2)/MAX_INIT_TEMP_ITERATIONS)
        < (INITIAL_ACCEPT_RATIO+INITIAL_ACCEPT_RANGE))
        CurTemp *= TEMP_INCREASE_FACTOR;
    else
    {
        fprintf(stderr,".\n");
        fprintf(stderr,"Done GetInitialTemp\n");
        return(CurTemp);
    }
    M1 = 0;
    M2 = 0;

    if (PrettyOutput/19)
    {
        fprintf(stderr,".\n");
        PrettyOutput=0;
    }
    else
    {
        fprintf(stderr,". ");
        PrettyOutput++;
    }
}/*end while*/
}/*end of function GetInitialTemp*/
```

```
/************************************************************************/
/*                  Function FillContoursRecursive*/
/*                                  */
/*  PURPOSE: This function uses recursion to "flood" a closed area in*/
/* a certain slice of the 3D array with a given value.*/
/*  INPUTS: The current position in the array, the array*/
/*  and the dimensions of the array are all inputs to the function*/
/*  OUTPUTS: None           */
/*                                  */
/************************************************************************/
void FillContoursRecursive(DoseArea *Array,int x,int y,int x_dim, int y_dim)
{

    if ((Array +x*(x_dim) + y) -> Type == EXTERNAL) /* if the position is currently "EXTERNAL" (ie:
'255') */
    {
        (Array +x*(x_dim) + y) -> Type = TUMOUR; /* fill the current position with the value */

        /* fill the area to the left, top, right, & bottom */
        if (x > 0) FillContoursRecursive(Array,x-1,y,x_dim,y_dim);
        if (y > 0) FillContoursRecursive(Array,x,y-1,x_dim,y_dim);
        if (x < x_dim-1) FillContoursRecursive(Array,x+1,y,x_dim,y_dim);
        if (y < y_dim-1) FillContoursRecursive(Array,x,y+1,x_dim,y_dim);
```

```
    }
    return;
}/*end of FillContoursRecursive*/



/*************************************************************************/
/*      Function: PlotOutputDose*/
/*                                */
/*  Purpose: This function will makd a PGM file using the  */
/*   source locations and tumour locations to find the dose */
/*  distribution.              */
/*  Inputs: Trial Number for output filename, linked list of sources*/
/*  Outputs: 1 if there is an error in the function*/
/*          else 0.               */
/*                                */
/*************************************************************************/
int PlotOutputDose(int Trial, PS **PSList)
{
int i=0,j=0;/* for the for loops */
PS *TempPS=NULL;/* Temp to run through the linked list */
char *FileName=NULL;/* The output filename */
FILE *OutputFile=NULL;/* the output file handle*/
float Dose=0.0f;
float Min=0.f;

#if DEBUG_PLOT_DOSE
fprintf(stderr,"In the PlotOutputDose function\n");
#endif

FileName = (char *)malloc(sizeof("OUTPUT/OutputDose      "));
sprintf(FileName,"OUTPUT/OutputDose%d.dat",Trial);
OutputFile = fopen(FileName,"w");
if (!OutputFile)
{
    fprintf(stderr,"Cannot open file %s for output, exiting PlotOutputDose\n",FileName);
    return(1);
}
free(FileName);

Min = 99999.9f;
for (i=0;i<xSize;i++)
{
    for (j=0;j<ySize;j++)
    {
        TempPS = *PSList;
        while (TempPS)
        {
            if ((TempPS -> x == i) && (TempPS -> y == j))
                Dose += TempPS -> DwellTime;
            else
                Dose += TempPS -> DwellTime/sqr(distance(i,j,TempPS->x,TempPS->y));

            TempPS = TempPS -> NextPS;
        }
        if (((Area + ySize*i + j) -> Type == TUMOUR)&&(Dose<Min))
            Min = Dose;

        if (((Area + ySize*i + j) -> Type == TUMOUR))
            fprintf(OutputFile,"%f ",Dose);
        Dose = 0.0f;
    }/*end for*/
    fprintf(OutputFile,"\n");
}/*end for*/
fclose(OutputFile);

/*now store the minimum in a file so that we can use it in the plotiso file*/
FileName = (char *)malloc(sizeof("OUTPUT/MinDose      "));
sprintf(FileName,"OUTPUT/MinDose%d.dat",Trial);
OutputFile = fopen(FileName,"w");
if (!OutputFile)
```

```
{
    fprintf(stderr,"Cannot open file %s for output, exiting PlotOutputDose\n",FileName);
    return(1);
}
free(FileName);
fprintf(OutputFile,"%f\n",Min);
fclose(OutputFile);
return(0);
}/*end function PlotOutputDose*/


/*************************************************************************/
/*      Function: MakeOutputPGM*/
/*                                */
/*  Purpose: This function will make a PGM file using the  */
/*   source locations and tumour locations */
/*  Inputs: None               */
/*  Outputs: 1 if there is an error in the function*/
/*          else 0.               */
/*                                */
/*************************************************************************/
int MakeOutputPGM(int Trial)
{
FILE *OutputFile=NULL;
PS *TempPS=NULL;
int i=0,j=0;
char *FileName=NULL;
DoseArea *TempArea=NULL;



    FileName = (char *)malloc(sizeof("OUTPUT/SourceConfig       "));
    sprintf(FileName,"OUTPUT/SourceConfig%d.pgm",Trial);
    OutputFile = fopen(FileName,"w");
    if (!OutputFile)
    {
        fprintf(stderr,"Cannot open file %s for output, exiting MakeOutputPGM\n",FileName);
        return(1);
    }
    free(FileName);

// Make the area to draw:
    if (!Area)
    {
        fprintf(stderr,"There is no area to draw in MakeOutputPGM, exiting!\n");
        return(1);
    }

    TempArea = (DoseArea *)malloc(xSize*ySize*sizeof(DoseArea));
    for (i=0;i<xSize;i++)
    {
        for (j=0;j<ySize;j++)
        {
            (TempArea+i*ySize+j)->Dose = (Area+i*ySize+j)->Dose;
            (TempArea+i*ySize+j)->Type = (Area+i*ySize+j)->Type;
        }
    }

// Put in the point sources:
    TempPS = FirstPS;
    if (!TempPS)
    {
        fprintf(stderr,"There are no point sources to draw in MakeOutputPGM\n");
        return(1);
    }

    while (TempPS)
    {
        (TempArea +(TempPS -> x) * ySize + TempPS -> y) -> Type = POINT_SOURCE;
        TempPS = TempPS -> NextPS;
    }
    fprintf(OutputFile,"P2\n");/* print file headers to the .pgm file */
```

```
    fprintf(OutputFile,"# CREATOR: XV Version 3.00  Rev: 3/30/93\n");
    fprintf(OutputFile,"%d %d\n",xSize,ySize);
    fprintf(OutputFile,"255\n");

    for (i = 0; i < xSize*ySize;i++)
    {
        if ((i % 17) == 0)/* we must print 18 points per line, hence the mod 17 */
        {
            fprintf(OutputFile,"\n%3d",(TempArea+i)->Type); /*start a new line*/
        }
        else
        {
            fprintf(OutputFile,"%4d",(TempArea+i)->Type);
        }
    }

    fclose(OutputFile);
    free(TempArea);
    return(0);
}/*end of function MakeOutputPGM*/



/*************************************************************************/
/*      Function: CreatAreaMap*/
/*                                */
/*  Purpose: This function will load and fill the tumour from file */
/*   The return value is 1 if there was an error during the */
/*   processing otherwise it is 0 */
/*  Inputs: None              */
/*                                */
/*************************************************************************/
int CreateAreaMap(char Name[128])
{
FILE *InputFile=NULL;/*Input file pointer*/
char name[128];/*For reading in from the input file*/
int i=0;        /*To parse input from file*/
int j=0;    /*To parse input from file*/
int value=0;/*Input from the file*/


#if DEBUG_CREATE_AREA
    fprintf(stderr,"In CreateAreaMap\n");
#endif
    fprintf(stderr,"\n\n\tOpening file %s for input....\n",Name);
    if (InputFile)
        fclose(InputFile);
    if ((InputFile = fopen(Name,"r")) == NULL)
    {
        fprintf(stderr,"\tInput file not found exiting\n\n");
        return(1);
    }
    else
        fprintf(stderr,"\tFile found, opened for input.\n\n");

/* Now parse input file to load in the tumour */
fgets(name,128,InputFile);/* P2 */
fgets(name,128,InputFile);/* # CREATOR: XV Version 3.01  Rev: 3/30/93 */
/* now get the x and y dimensions of the file */
fscanf(InputFile,"%d %d\n",&xSize,&ySize);

fgets(name,128,InputFile);/* 255 */

/*malloc the size of the array for the input*/
Area = (DoseArea *)malloc(xSize*ySize*sizeof(DoseArea));

/* now get all of the input */
GlobalTumourCount=0;
for (i=0;i<xSize;i++)
{
```

```
    for (j=0;j<ySize;j++)
    {
        fscanf(InputFile,"%d",&value);
        if (value==EXTERNAL)/*save as external location*/
        {
            (Area + i*ySize + j) -> Dose = 0.0f;
            (Area + i*ySize + j) -> Type = EXTERNAL;
        }
        else if (value == MOVABLE_LOCATION) /*save as external movable location*/
        {
            (Area + i*ySize + j) -> Dose = 0.0f;
            (Area + i*ySize + j) -> Type = MOVABLE_LOCATION;
#if ALLOW_SOURCES_OUTSIDE
    GlobalTumourCount++;
#endif
        }
        else if (value == PERIPHERY)
        {
            (Area + i*ySize + j) -> Dose = 0.0f;
            (Area + i*ySize + j) -> Type = PERIPHERY;
            GlobalTumourCount++;
        }
        else/* it is a tumour point */
        {
            (Area + i*ySize + j) -> Dose = 0.0f;
            (Area + i*ySize + j) -> Type = TUMOUR;
            GlobalTumourCount++;
        }
    }/*end for j*/
}/*end for i*/

/* Close the input file */
fclose(InputFile);

#if DEBUG_CREATE_AREA
InputFile = fopen("CreateAreaMap.pgm","w");
fprintf(InputFile,"P2\n");
fprintf(InputFile,"# CREATOR: XV Version 3.01  Rev: 3/30/93\n");
fprintf(InputFile,"%i %i\n",xSize,ySize);
fprintf(InputFile,"255\n");
for (i=0;i<xSize;i++)
{
    for (j=0;j<ySize;j++)
    {
        fprintf(InputFile,"%d ",(Area + i*ySize + j) -> Type);
    }
    fprintf(InputFile,"\n");
}
fclose(InputFile);
fprintf(stderr,"Done CreateAreaMap\n");
#endif

return(0);
}/*end of CreateAreaMap*/
```

```
/*************************************************************************/
/*      Function: SeedRandom()*/
/*                                */
/* Purpose: This function will seed the C randon number generator */
/* IT MUST ONLY BE CALLED ONCE DURING A SINGLE TIME STEP */
/* The return value is 1 if there is an error else, it is */
/* 0.                                */
/* Input: None.                */
/* Output: None.           */
```

```
/*                                */
/*********************************************************************/
int SeedRandom()
{
long ltime=0;
int stime=0;

/* get current calender time */
ltime = time(NULL);
stime = (unsigned) ltime/2;
srand(stime);
return(0);
}/*end of SeedRandom*/
```

```
/*********************************************************************/
/*                                */
/*      Function : CheckCostFnct*/
/* Purpose: This is a function that can be called to "see" */
/* how the cost function is working. The return */
/* value is 1 if there was a problem, otherwise */
/* it is 0                        */
/* Input : None                   */
/* Output : int if no problems occured, and generates a file cost.dat*/
/*                                */
/*********************************************************************/
int CheckCostFnct()
{
FILE *CostFile=NULL;
int i=0;
int j=0;
float cost=0.0f;
float MinCost=32000.0f;
int MinX=0,MinY=0;

CostFile = fopen("cost.dat","w");

#if DEBUG_CHECK_COST
    fprintf(stderr,"In CheckCostFnct\n");
#endif

for (i=0;i<xSize;i++)
{
    for (j=0;j<ySize;j++)
    {
        /* if ((Area+i*ySize + j) -> Type == EXTERNAL)
        {
            fprintf(CostFile,"0.0 ");
            continue;
        } */

        FirstPS = (PS *)malloc(sizeof(PS));
        FirstPS -> x = i;
        FirstPS -> y = j;
        FirstPS -> DwellTime = DEFAULT_DWELL_TIME;
        FirstPS -> NextPS = NULL;

        cost = CalculateCost(FirstPS);
        #if DEBUG_CHECK_COST
        if (cost<MinCost)
        {
            MinCost = cost;
            MinX = i;
            MinY = j;
        }
        #endif
        fprintf(CostFile,"%f ",cost);
```

```
        FreePS(&FirstPS);
    }
    fprintf(CostFile,"\n");
#if DEBUG_CHECK_COST
    fprintf(stderr,"%i\n",i);
#endif
}

fclose(CostFile);
#if DEBUG_CHECK_COST
    fprintf(stderr,"Done CheckCostFnct Min of %f at x=%d and y=%d\n",MinCost,MinX,MinY);
#endif
return(0);
}/*end of function CheckCostFnct*/



/*****************************************/
/*  MAIN LINE PROGRAM*/
/*****************************************/
int main(int argc,char *argv[])
{
time_t lt;
struct tm *ptr=NULL;
int trial=0;
int NUMBER_OF_SOURCES = DEFAULT_NUMBER_OF_SOURCES;/* variable for the number of sources */


fprintf(stderr,"\n\nThis is a program for optimizing point sources using Simulated Annealing.\n");
fprintf(stderr,"Sample input is: 'SimAnn.exe 0' 'filename.pgm 1' '#sources 2' 'dwell time 3'\n 'pre-
scribed dose 4' 'internal weight 5' 'external weight 6'\n 'percentage of prescribed to external 7'
'reduction factor 8' 'iterations 9' 'stop iterations 10' 'with in %% 11' 'trial # 12'\n\n");



if (!argv[2])
    NUMBER_OF_SOURCES = DEFAULT_NUMBER_OF_SOURCES;
else
    NUMBER_OF_SOURCES = (int)atoi(argv[2]);

if (!argv[3])
    DWELL_TIME = DEFAULT_DWELL_TIME;
else
    DWELL_TIME = (float)atof(argv[3]);

if (!argv[4])
    PRESCRIBED_DOSE = DEFAULT_PRESCRIBED_DOSE;
else
    PRESCRIBED_DOSE = (float)atof(argv[4]);

if (!argv[5])
    INTERNAL_WEIGHT = DEFAULT_INTERNAL_WEIGHT;
else
    INTERNAL_WEIGHT = (float)atof(argv[5]);

if (!argv[6])
    EXTERNAL_WEIGHT = DEFAULT_EXTERNAL_WEIGHT;
else
    EXTERNAL_WEIGHT = (float)atof(argv[6]);

if (!argv[7])
    EXTERNAL_FACTOR = DEFAULT_EXTERNAL_FACTOR;
else
    EXTERNAL_FACTOR = (float)atof(argv[7]);

if (!argv[8])
    TEMP_REDUCTION_FACTOR = DEFAULT_TEMP_REDUCTION_FACTOR;
else
    TEMP_REDUCTION_FACTOR = (float)atof(argv[8]);

if (!argv[9])
    MAX_ITERATIONS = DEFAULT_MAX_ITERATIONS;
```

```
else
    MAX_ITERATIONS = (int)atoi(argv[9]);

if (!argv[10])
    STOP_COUNT = DEFAULT_STOP_COUNT;
else
    STOP_COUNT = (int)atoi(argv[10]);

if (!argv[11])
    WITHIN_FACTOR = DEFAULT_WITHIN_FACTOR;
else
    WITHIN_FACTOR = (float)atof(argv[11]);

if (!argv[12])
    trial = 1;
else
    trial = (int)atoi(argv[12]);


/* loads in the tumour file(s) _and_ fills them */
if (CreateAreaMap(argv[1]))
{
    fprintf(stderr,"Error in CreateAreaMap exiting.\n");
    exit(0);
}



#if DEBUG_PLOT_DOSE
    InsertSources(NUMBER_OF_SOURCES);
    PlotOutputDose(trial,&FirstPS);
    FreePS(&FirstPS);
    fprintf(stderr,"Done checking Dose plotting\n");
    exit(0);
#endif



/* Seeds the C random number generator ONLY DO THIS ONCE */
if (SeedRandom())
{
    fprintf(stderr,"Error in SeedRandom exiting.\n");
    exit(0);
}

#if DEBUG_CHECK_COST
/* generate 3D map of cost function */
if (CheckCostFnct())
{
    fprintf(stderr,"Error in CheckCostFnct exiting.\n");
    exit(0);
}
#endif

fprintf(stderr,"External factor: %f\n",EXTERNAL_FACTOR);
lt = time(NULL);
ptr = localtime(&lt);
fprintf(stderr,"Starting SimAnn time is: %s\n\n",asctime(ptr));

if (SimulatedAnnealing(GetInitialTemp(NUMBER_OF_SOURCES),trial,NUMBER_OF_SOURCES))
{
    fprintf(stderr,"Error in SimulatedAnnealing exiting.\n");
    exit(0);
}

lt = time(NULL);
ptr = localtime(&lt);
fprintf(stderr,"Done SimAnn time: %s\n\n",asctime(ptr));

if (MakeOutputPGM(trial))
{
    fprintf(stderr,"Error in MakeOutputPGM exiting.\n");
```

```
    exit(0);
}


if (PlotOutputDose(trial,&FirstPS))
{
    fprintf(stderr,"Error in PlotOutputDose exiting.\n");
    exit(0);
}

/*Free global memory that was malloced*/
free(Area);
FreePS(&FirstPS);
}/*end of main line*/
```

E.31 MainFrm.cpp

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


#include "stdafx.h"
#include <math.h>

#include "Isocurve.h"
#include "ViewDataDlg.h"
#include "AddTypeDlg.h"

#include "MainFrm.h"
#include "IsocurveDoc.h"
#include "IsocurveView.h"

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

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    ON_WM_CONTEXTMENU()
    ON_COMMAND(ID_CONTEXT_VIEW, OnContextViewData)
    ON_COMMAND(ID_CONTEXT_ADD, OnContextAdd)
    ON_COMMAND(ID_CONTEXT_DELETE, OnContextDelete)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
    // TODO: add member initialization code here
    m_pContextMenu = new CMenu();
```

```
    m_pContextMenu->LoadMenu(IDR_CONTEXTMENU);
    m_IsDown = FALSE;
}

CMainFrame::~CMainFrame()
{
    delete m_pContextMenu;
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
        | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;      // fail to create
    }

    // TODO: Delete these three lines if you don't want the toolbar to
    //  be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CMDIFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CMDIFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers


void CMainFrame::OnContextMenu(CWnd* pWnd, CPoint point)
{
    CIsocurveDoc * pDoc = GetDocument();
```

```
    if (!pDoc)
        return;

    // now if this is the right type of document for this context menu (IsocurveDoc that is)
    if (pDoc->IsKindOf(RUNTIME_CLASS(CIsocurveDoc)))
    {
        //check that there is an instance loaded
        if (pDoc->m_LoadedFile)
        {
            CMenu * pMenu;
            pMenu = m_pContextMenu->GetSubMenu(0);
            pMenu->TrackPopupMenu(TPM_LEFTALIGN|TPM_RIGHTBUTTON,point.x,point.y,this,NULL);
            m_XMousePos = point.x;
            m_YMousePos = point.y;
        }//end if loaded file
        else
            AfxMessageBox("there is none loaded");
    }
    else
    {
        AfxMessageBox("Wrong type of document");
        return;
    }
}

void CMainFrame::OnContextViewData()
{
    CIsocurveDoc * pDoc = GetDocument();

    if (!pDoc)
        return;

    // now if this is the right type of document for this context menu (IsocurveDoc that is)
    if (!pDoc->IsKindOf(RUNTIME_CLASS(CIsocurveDoc)))
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    pDoc->CalculateDose();
    int value = pDoc->m_Area[i*pDoc->m_ySize+j].Type;
    float dose = pDoc->m_Area[i*pDoc->m_ySize+j].Dose;

//bring up the dialog box with the data in it:
    CViewDataDlg dlg;

    dlg.m_X = i;
    dlg.m_Y = j;
    dlg.m_Dose = dose;
    switch (value)
    {
        case(0):
            dlg.m_Type = "Tumour";
            break;
        case(40):
            dlg.m_Type = "Periphery";
            break;
        case(175):
            dlg.m_Type = "Point Source";
            break;
        case(255):
            dlg.m_Type = "External";
            break;
        default:
            dlg.m_Type = "Unknown";
            break;
    }
    dlg.DoModal();
}

void CMainFrame::FindAreaIndex(int *i, int *j)
{
    CIsocurveDoc * pDoc = GetDocument();
```

```
    if (!pDoc)
        return;

    // now it is in the list we need to add it to the image
    // now if this is the right type of document for this context menu (IsocurveDoc that is)
    int DocSizeX,DocSizeY;
    if (pDoc->IsKindOf(RUNTIME_CLASS(CIsocurveDoc)))
    {
        DocSizeX = pDoc->m_xSize;
        DocSizeY = pDoc->m_ySize;
    }
    else
        return;

    CPoint TempPoint(m_XMousePos,m_YMousePos);
    // Get the MainFrame
    CMDIFrameWnd *pFrame;
    pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;
    // Get the active MDI child window.
    CMDIChildWnd *pChild;
    if (!pFrame)
        return;
    pChild = (CMDIChildWnd *) pFrame->GetActiveFrame();// or CMDIChildWnd *pChild = pFrame->MDIGetAc-
tive();
    pChild->ScreenToClient(&TempPoint);

    *i = (int)round(((float)TempPoint.x/(float)m_WinSizeX)*(float)(DocSizeX))-1;
    *j = (int)round(((float)TempPoint.y/(float)m_WinSizeY)*(float)(DocSizeY))-1;
    //CString s;
    //s.Format("%d %d %d %d %d %d\n",WinSizeX,WinSizeY,DocSizeX,DocSizeY,*i,*j);
    //AfxMessageBox(s);
}

int CMainFrame::round(float x)
{
    //return((int)ceil(x-0.5));
    return((int)ceil(x));
}

CIsocurveDoc * CMainFrame::GetDocument()
{
    CMDIFrameWnd *pFrame;
    pFrame = (CMDIFrameWnd*)AfxGetApp()->m_pMainWnd;

    // Get the active MDI child window.
    CMDIChildWnd *pChild;
    if (!pFrame)
        return(NULL);


    pChild = (CMDIChildWnd *) pFrame->GetActiveFrame();// or CMDIChildWnd *pChild = pFrame->MDIGetAc-
tive();

    // Get the active view attached to the active MDI child window.
    CIsocurveView *pView;
    if (!pChild)
        return(NULL);

    pView = (CIsocurveView *) pChild->GetActiveView();

    if (!pView)
        return(NULL);

    if (pView->IsKindOf(RUNTIME_CLASS(CIsocurveView)))
    {
        m_WinSizeX = pView->m_cx;
        m_WinSizeY = pView->m_cy;
    }
    else
        return(NULL);

    return((CIsocurveDoc *) pView->GetDocument());
```

```
}

void CMainFrame::OnContextAdd()
{
    CAddTypeDlg dlg;

    dlg.m_AddType = 0;
    if (dlg.DoModal()==IDOK)
    {
        switch(dlg.m_AddType)
        {
            case(0):
                AddSource();
                break;
            case(1):
                AddTumor();
                break;
            case(2):
                AddPeriphery();
                break;
            case(3):
                AddOutside();
                break;
            case(4):
                AddDesiredPS();
                break;
            default:
                break;
        }
    }
}

void CMainFrame::OnContextDelete()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    //find if it is a pointsource first then if so delete
    if (pDoc->m_Area[i*pDoc->m_ySize+j].Type == 175)
    {
        // delete it
        pDoc->DeletePointSource(i,j);
        // find what user wants to set it to
        //dlg;
        pDoc->m_Area[i*pDoc->m_ySize+j].Type = 0;
        pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
        //redraw the scene
        pDoc->UpdateAllViews(NULL);
        //recalculate the dose
        pDoc->CalculateDose();
    }
    else
        AfxMessageBox("There is not a point source there to remove!\n");

}

void CMainFrame::AddSource()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    // add it to linked list of point sources in document
    pDoc->AddPointSource(i,j,1);
    // change the area type
    pDoc->m_Area[i*pDoc->m_ySize+j].Type = 175;
    pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
```

```
    //redraw the scene
    pDoc->UpdateAllViews(NULL);
    //recalculate the dose
    pDoc->CalculateDose();
}

void CMainFrame::AddTumor()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    // change the area type
    pDoc->m_Area[i*pDoc->m_ySize+j].Type = 0;
    pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
    //redraw the scene
    pDoc->UpdateAllViews(NULL);
    //recalculate the dose
    pDoc->CalculateDose();
}

void CMainFrame::AddPeriphery()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    // change the area type
    pDoc->m_Area[i*pDoc->m_ySize+j].Type = 40;
    pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
    //redraw the scene
    pDoc->UpdateAllViews(NULL);
    //recalculate the dose
    pDoc->CalculateDose();
}

void CMainFrame::AddOutside()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    // change the area type
    pDoc->m_Area[i*pDoc->m_ySize+j].Type = 255;
    pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
    //redraw the scene
    pDoc->UpdateAllViews(NULL);
    //recalculate the dose
    pDoc->CalculateDose();
}

void CMainFrame::AddDesiredPS()
{
    CIsocurveDoc * pDoc = GetDocument();
    if (!pDoc)
        return;

    int i,j;
    FindAreaIndex(&i,&j);
    // add a desired ps
    pDoc->AddDesiredPointSource(i,j,1);
    // change the area type
    pDoc->m_Area[i*pDoc->m_ySize+j].Type = 175;
    pDoc->m_Area[i*pDoc->m_ySize+j].Dose = 0.0;
    //redraw the scene
    pDoc->UpdateAllViews(NULL);
```

```
    //recalculate the dose
    pDoc->CalculateDose();
}
```

E.32 MainFrm.h

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

#if !defined(AFX_MAINFRM_H__E8F0A3A8_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_MAINFRM_H__E8F0A3A8_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_
#include "IsocurveDoc.h"

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

class CMainFrame : public CMDIFrameWnd
{
    DECLARE_DYNAMIC(CMainFrame)
public:
    CMainFrame();

// Attributes
public:

// Operations
public:

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

// Implementation
public:
    void AddDesiredPS(void);
    BOOL m_IsDown;
    void AddOutside();
    void AddPeriphery();
    void AddTumor();
    void AddSource();
    void FindAreaIndex(int *i,int *j);
    int m_YMousePos;
    int m_XMousePos;
    virtual ~CMainFrame();
    int m_WinSizeY;
    int m_WinSizeX;

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
    CStatusBar  m_wndStatusBar;
    CToolBar    m_wndToolBar;

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnContextMenu(CWnd* pWnd, CPoint point);
    afx_msg void OnContextViewData();
    afx_msg void OnContextAdd();
    afx_msg void OnContextDelete();
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
private:
    CMenu * m_pContextMenu;
    int CMainFrame::round(float x);
    CIsocurveDoc * GetDocument();
};

/////////////////////////////////////////////////////////////////////////////

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

#endif // !defined(AFX_MAINFRM_H__E8F0A3A8_4347_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```

E.33 MultFactorDlg.cpp

```
// MultFactorDlg.cpp : implementation file
//

#include "stdafx.h"
#include "isocurve.h"
#include "MultFactorDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CMultFactorDlg dialog


CMultFactorDlg::CMultFactorDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CMultFactorDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CMultFactorDlg)
    m_MultFact = 0.0f;
    //}}AFX_DATA_INIT
}


void CMultFactorDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CMultFactorDlg)
    DDX_Text(pDX, IDC_MULTFACT, m_MultFact);
    DDV_MinMaxFloat(pDX, m_MultFact, 1.e-004f, 100.f);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CMultFactorDlg, CDialog)
    //{{AFX_MSG_MAP(CMultFactorDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMultFactorDlg message handlers
```

E.34 MultFactorDlg.h

```
#if !defined(AFX_MULTFACTORDLG_H__A31CE135_C9E0_11D3_8088_0090276F4183__INCLUDED_)
#define AFX_MULTFACTORDLG_H__A31CE135_C9E0_11D3_8088_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// MultFactorDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMultFactorDlg dialog

class CMultFactorDlg : public CDialog
{
// Construction
public:
    CMultFactorDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CMultFactorDlg)
    enum { IDD = IDD_MULTFACT };
    floatm_MultFact;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMultFactorDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CMultFactorDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_MULTFACTORDLG_H__A31CE135_C9E0_11D3_8088_0090276F4183__INCLUDED_)
```

E.35 NewShapeDlg.cpp

```
// NewShapeDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "NewShapeDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CNewShapeDlg dialog
```

```
CNewShapeDlg::CNewShapeDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CNewShapeDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CNewShapeDlg)
    m_CanvasX = 0;
    m_CanvasY = 0;
    m_Circle = 0;
    m_SquareX = 0;
    m_SquareY = 0;
    m_CrclRad = 0;
    //}}AFX_DATA_INIT

    //IDC_RADIUS->Enabled = FALSE;
}


void CNewShapeDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CNewShapeDlg)
    DDX_Control(pDX, IDC_SQUAREY, m_SqrYCtrl);
    DDX_Control(pDX, IDC_SQUAREX, m_SqrXCtrl);
    DDX_Control(pDX, IDC_SQRPROP, m_SqrProp);
    DDX_Control(pDX, IDC_SQRY, m_SqrY);
    DDX_Control(pDX, IDC_SQRX, m_SqrX);
    DDX_Control(pDX, IDC_RADIUS, m_Radius);
    DDX_Control(pDX, IDC_RAD, m_Rad);
    DDX_Control(pDX, IDC_CRCLPROP, m_CrclProp);
    DDX_Text(pDX, IDC_CANVASX, m_CanvasX);
    DDV_MinMaxInt(pDX, m_CanvasX, 10, 1000);
    DDX_Text(pDX, IDC_CANVASY, m_CanvasY);
    DDV_MinMaxInt(pDX, m_CanvasY, 10, 1000);
    DDX_Radio(pDX, IDC_RADIO1, m_Circle);
    DDX_Text(pDX, IDC_SQUAREX, m_SquareX);
    DDV_MinMaxInt(pDX, m_SquareX, 10, 1000);
    DDX_Text(pDX, IDC_SQUAREY, m_SquareY);
    DDV_MinMaxInt(pDX, m_SquareY, 10, 1000);
    DDX_Text(pDX, IDC_RADIUS, m_CrclRad);
    DDV_MinMaxInt(pDX, m_CrclRad, 2, 500);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CNewShapeDlg, CDialog)
    //{{AFX_MSG_MAP(CNewShapeDlg)
    ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
    ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CNewShapeDlg message handlers

void CNewShapeDlg::OnRadio1() //square
{
    EnableSquare(TRUE);
    EnableCircle(FALSE);
}

void CNewShapeDlg::OnRadio2() //circle
{
    EnableSquare(FALSE);
    EnableCircle(TRUE);
}

void CNewShapeDlg::EnableSquare(BOOL enable)
{
    m_SqrProp.EnableWindow(enable);
    m_SqrX.EnableWindow(enable);
    m_SqrY.EnableWindow(enable);
    m_SqrXCtrl.EnableWindow(enable);
```

```
    m_SqrYCtrl.EnableWindow(enable);
}

void CNewShapeDlg::EnableCircle(BOOL enable)
{
    m_CrclProp.EnableWindow(enable);
    m_Radius.EnableWindow(enable);
    m_Rad.EnableWindow(enable);
}
```

E.36 NewShapeDlg.h

```
#if !defined(AFX_NEWSHAPEDLG_H__62F67C22_A805_11D3_8082_0090276F4183__INCLUDED_)
#define AFX_NEWSHAPEDLG_H__62F67C22_A805_11D3_8082_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// NewShapeDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CNewShapeDlg dialog

class CNewShapeDlg : public CDialog
{
// Construction
public:
    void EnableCircle(BOOL enable);
    void EnableSquare(BOOL enable);
    CNewShapeDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CNewShapeDlg)
    enum { IDD = IDD_NEWSHAPE };
    CEditm_SqrYCtrl;
    CEditm_SqrXCtrl;
    CButtonm_SqrProp;
    CStaticm_SqrY;
    CStaticm_SqrX;
    CEditm_Radius;
    CStaticm_Rad;
    CButtonm_CrclProp;
    int     m_CanvasX;
    int     m_CanvasY;
    int     m_Circle;
    int     m_SquareX;
    int     m_SquareY;
    int     m_CrclRad;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNewShapeDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CNewShapeDlg)
    afx_msg void OnRadio1();
    afx_msg void OnRadio2();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

```
};

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

#endif // !defined(AFX_NEWSHAPEDLG_H__62F67C22_A805_11D3_8082_0090276F4183__INCLUDED_)
```

E.37 OutputPropsDlg.cpp

```
// OutputPropsDlg.cpp : implementation file
//

#include "stdafx.h"
#include "isocurve.h"
#include "OutputPropsDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// COutputPropsDlg dialog


COutputPropsDlg::COutputPropsDlg(CWnd* pParent /*=NULL*/)
    : CDialog(COutputPropsDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(COutputPropsDlg)
    m_Manhattan = 0;
    m_NumInst = 0;
    m_Continous = 0;
    m_New = 1;
    //}}AFX_DATA_INIT
}


void COutputPropsDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(COutputPropsDlg)
    DDX_Text(pDX, IDC_MANHATTAN, m_Manhattan);
    DDV_MinMaxInt(pDX, m_Manhattan, 0, 1000);
    DDX_Text(pDX, IDC_NUMINST, m_NumInst);
    DDV_MinMaxInt(pDX, m_NumInst, 0, 1000);
    DDX_Radio(pDX, IDC_CONT, m_Continous);
    DDX_Radio(pDX, IDC_NEW, m_New);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(COutputPropsDlg, CDialog)
    //{{AFX_MSG_MAP(COutputPropsDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// COutputPropsDlg message handlers
```

E.38 OutputPropsDlg.h

```
#if !defined(AFX_OUTPUTPROPSDLG_H__10D6887F_B23A_11D3_8082_0090276F4183__INCLUDED_)
#define AFX_OUTPUTPROPSDLG_H__10D6887F_B23A_11D3_8082_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// OutputPropsDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// COutputPropsDlg dialog

class COutputPropsDlg : public CDialog
{
// Construction
public:
    COutputPropsDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(COutputPropsDlg)
    enum { IDD = IDD_OUTPUTPROPS };
    int     m_Manhattan;
    int     m_NumInst;
    int     m_Continous;
    int     m_New;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(COutputPropsDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(COutputPropsDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_OUTPUTPROPSDLG_H__10D6887F_B23A_11D3_8082_0090276F4183__INCLUDED_)
```

E.39 Ps.c

```
#include <stdlib.h>
#include <stdio.h>
#include "globals.h"


/*************************************************************************/
/*                  Function: InsertSources*/
/*                                  */
/*  Purpose : This function will insert the number of sources that*/
/*  are used in a single solution. The sources are inserted rand-*/
/*  omly.                           */
/*  Input : None;               */
```

```
/*  Output : The global variable FirstPS will point to a linked list*/
/*  of point sources.      */
/*                                  */
/*************************************************************************/
void InsertSources(int num_sources)
{
int i=0;
PS *TempPS=NULL;
PS *CurPS=NULL;

#if DEBUG_INSERT_SOURCES
    fprintf(stderr,"Starting InsertSources\n");
#endif


/* check to make sure that there are not too many sources to insert*/
if (num_sources >= GlobalTumourCount)
{
    fprintf(stderr,"There are too many sources. I have to quit there is a problem.\n");
    exit(0);
}

for (i=0;i<num_sources;i++)
{
#if DEBUG_INSERT_SOURCES
    fprintf(stderr,"insert source: %i\n",i);
#endif


    MakePS(&TempPS);
    if (!FirstPS)
        FirstPS = TempPS;
    else
        CurPS -> NextPS = TempPS;
    CurPS = TempPS;
}

#if DEBUG_INSERT_SOURCES
    TempPS = FirstPS;
    fprintf(stderr,"In InsertSources\n");
    while (TempPS)
    {
        fprintf(stderr,"Pointsource x: %i  y: %i\n",TempPS -> x,TempPS -> y);
        TempPS = TempPS -> NextPS;
    }
    fprintf(stderr,"Done insert point sources\n");
#endif
}/*end of function InsertSources*/




/*************************************************************************/
/*                  Function: MoveSources*/
/*                                  */
/*  Purpose : The function will move sources within the tumour */
/*  volume, by an amount related to the current temperature.*/
/*  Inputs:The initial and current temperatures in the sim.*/
/*  Outputs:An 1 if there is a problem else a 0.*/
/*                                  */
/*************************************************************************/
int MoveSources(float InitialTemp,float CurTemp)
{
float TempRand=0.0f;/* To hold the random number generated */
int WhichSource=0;/* which source is going to move */
int i=0;/* for loop counter*/
int NumSources=0;
```

```
//if ((InitialTemp>0.0f) && (CurTemp>0.0f))
//{
    //MultFactor = round(MAX_SOURCE_MOVE - MAX_SOURCE_MOVE*((InitialTemp - CurTemp)/InitialTemp));
//}
//else
//{
//  fprintf(stderr,"In MoveSources and there is a problem with temperatures.\n");
//  return(1);
//}


if ((MOVE_PERCENT + ADD_PERCENT + DELETE_PERCENT < 1.0f) || (MOVE_PERCENT + ADD_PERCENT + DELETE_PERCENT
> 1.0f))
{
    fprintf(stderr,"In MoveSources and the percentages do not add up, exiting\n");
    return(1);
}

// to get rid of the global variables for number of sources:
NumSources = NumPS(FirstPS);

/* this is for deciding wether to move or add */
TempRand = (float)rand()/RAND_MAX;
if (TempRand < MOVE_PERCENT)/* this is the code for moving sources */
{
    WhichSource=0;
    TempRand = (float)rand()/RAND_MAX;
    for (i=0;i<NumSources;i++)
    {
        if ((((float)(i)/NumSources) < TempRand) && (TempRand < (float)(i+1)/NumSources))
            WhichSource = i;
    }
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"\nMoving %d\n",NumSources);
    if (!FirstPS){ fprintf(stderr,"b/f move %p\n\a",FirstPS);exit(0);}
    PrintPS(FirstPS);
#endif
    if (NumSources && (NumSources < GlobalTumourCount))
        MovePS(WhichSource,FirstPS,&FirstNewPS);
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"New points are: %p\n",FirstNewPS);
    PrintPS(FirstNewPS);
    fprintf(stderr,"\n");
#endif
}/* end move */
else if (TempRand < MOVE_PERCENT + ADD_PERCENT)
{
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"\nAdding %d\n",NumSources);
    if (!FirstPS){ fprintf(stderr,"b/f add %p\n\a",FirstPS);exit(0);}
    PrintPS(FirstPS);
#endif

#if JUST_MOVE
        /* to find the lucky source to be moved */
        WhichSource=0;
        TempRand = (float)rand()/RAND_MAX;
        for (i=0;i<NumSources;i++)
        {
            if ((((float)(i)/NumSources) < TempRand) && (TempRand < (float)(i+1)/NumSources))
                WhichSource = i;
        }
        MovePS(WhichSource,FirstPS,&FirstNewPS);
#else
    if (NumSources < GlobalTumourCount)
        AddPS(FirstPS,&FirstNewPS);/* attach to the end of the list */
#endif
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"New points are: %p\n",FirstNewPS);
    PrintPS(FirstNewPS);
    fprintf(stderr,"\n");
```

```
#endif
}/* end else add */
else/* delete a source */
{
    /* to find the lucky source to be deleted */
    WhichSource=0;
    TempRand = (float)rand()/RAND_MAX;
    for (i=0;i<NumSources;i++)
    {
        if ((((float)(i)/NumSources) < TempRand) && (TempRand < (float)(i+1)/NumSources))
            WhichSource = i;
    }
    //make sure there are enough sources to delete


#if JUST_MOVE
    if (NumSources<GlobalTumourCount)
        MovePS(WhichSource,FirstPS,&FirstNewPS);
#else
#if !ALLOW_0_SOURCES
    if (NumSources>1)
#endif
        DeletePS(WhichSource,FirstPS,&FirstNewPS);/*delete specified source */
#endif
}/* end of else delete*/

#if DEBUG_MOVE_SOURCES
fprintf(stderr,"New Points are:\n");
PrintPS(FirstNewPS);
fprintf(stderr,"---\n");
#endif
return(0);
}/*end of function move sources*/

/************************************************************************/
/*          Function : FreePS*/
/*                                  */
/*  Purpose : To free a linked list of PointSources from the */
/*  simulation.                 */
/*  Input : The address of the first pointer.*/
/*  Output : None, but the memory that was malloced is freed*/
/*                                  */
/************************************************************************/
void FreePS(PS **list)
{
PS *temp;

temp = *list;
while (temp)
{
    *list = temp;
    temp = temp -> NextPS;
    free(*list);
    *list = NULL;
}
}/*end of function freePS*/


/************************************************************************/
/*          Function : SwapPS*/
/*                                  */
/*  Purpose : This function will swap the memory of two INTO one*/
/*  Input : The adresses of two PS linked lists.*/
/*  Output : None.              */
/*                                  */
/************************************************************************/
void SwapPS(PS **one, PS **two)
{
PS *temp=NULL;

temp = *one;
*one = *two;
```

```
*two = temp;

/* free what used to be one but is now two */
FreePS(two);
}/*end of function swap*/



/************************************************************************/
/*          Function : CopyPS*/
/*                                 */
/*  Purpose : This function will copy the memory of two INTO one*/
/*  Input : The adresses of two PS linked lists.*/
/*  Output : None.              */
/*                                 */
/************************************************************************/
void CopyPS(PS **one, PS *two)
{
PS *CurPS=NULL;
PS *CurNewPS=NULL;
PS *NewPS=NULL;

    CurPS = two;
    while (CurPS)
    {
        NewPS = (PS*)malloc(sizeof(PS));

        NewPS -> x = CurPS -> x;
        NewPS -> y = CurPS -> y;
        NewPS -> DwellTime = CurPS -> DwellTime;
        NewPS -> NextPS = NULL;

        if (!*one)
            *one = NewPS;
        else
            CurNewPS -> NextPS = NewPS;

        CurNewPS = NewPS;
        CurPS = CurPS -> NextPS; /* go to next source */
    }
}/*end of function swap*/



/************************************************************************/
/*          Function : PrintPS*/
/*                                 */
/*  Purpose : This function will print out a PS linked list*/
/*  Input : A PS linked list.*/
/*  Output : Text to stderr.*/
/*                                 */
/************************************************************************/
void PrintPS(PS *list)
{
PS *temp=NULL;
int element=0;

fprintf(stderr,"There are %d PS in the list\n",NumPS(list));
temp = list;
while (temp)
{
    //fprintf(stderr,"Structure-%d x:%d and y:%d\n",element,temp->x,temp->y);
    fprintf(stderr,"%d %d\n",temp->x,temp->y);
    element++;
    temp = temp -> NextPS;
}
}/*end of function PrintPS*/



/************************************************************************/
/*          Function : AddPS*/
```

```
/*                                */
/*  Purpose : This function will add a PS to the PS linked list*/
/*  Input : A PS linked list.*/
/*  Output : Text to stderr.*/
/*                                */
/*********************************************************************/
void AddPS(PS *original_list,PS **new_list)
{
PS *CurPS=NULL;
PS *CurNewPS=NULL;
PS *NewPS=NULL;

#if DEBUG_ADDPS
    fprintf(stderr,"In AddPS\n");
#endif

//need to get to the end of the list
    CurPS = original_list;
    while (CurPS)
    {
        NewPS = (PS*)malloc(sizeof(PS));

        NewPS -> x = CurPS -> x;
        NewPS -> y = CurPS -> y;
        NewPS -> DwellTime = CurPS -> DwellTime;
        NewPS -> NextPS = NULL;

        if (!*new_list)
            *new_list = NewPS;
        else
            CurNewPS -> NextPS = NewPS;

        CurNewPS = NewPS;
        CurPS = CurPS -> NextPS; /* go to next source */
    }
// now add one to the end of the list
    MakePS(&NewPS);
// just in case there are none in the list maybe try to add it to the start
    if (!*new_list)
        *new_list = NewPS;
    else
        CurNewPS -> NextPS = NewPS;

    CurNewPS = NewPS;
}/*end of function AddPS*/


/*********************************************************************/
/*          Function : DeletePS*/
/*                                */
/*  Purpose : This function will delete a PS from a PS linked list*/
/*  Input : A PS linked list.*/
/*  Output : Text to stderr.*/
/*                                */
/*********************************************************************/
void DeletePS(int index,PS *original_list,PS **new_list)
{
PS *CurPS=NULL;
PS *CurNewPS=NULL;
PS *NewPS=NULL;
int source_count=0;


    //fprintf(stderr,"look at me delete\n");
    CurPS = original_list;
    source_count=0;
    while (CurPS)
    {
        if (source_count == index) //delete this one
            CurPS = CurPS -> NextPS; /* go to next source */
        if (!CurPS)
            break;
```

```
        NewPS = (PS*)malloc(sizeof(PS));
        NewPS -> x = CurPS -> x;
        NewPS -> y = CurPS -> y;
        NewPS -> DwellTime = CurPS -> DwellTime;
        NewPS -> NextPS = NULL;

        if (!*new_list)
            *new_list = NewPS;
        else
            CurNewPS -> NextPS = NewPS;

        CurNewPS = NewPS;
        CurPS = CurPS -> NextPS; /* go to next source */
        source_count++;
    }
}/*end of function DeletePS*/


/*************************************************************************/
/*          Function : MakePS*/
/*                                */
/*  Purpose : This function will make a PS*/
/*  Input : A PS pointer.       */
/*  Output : Text to stderr.*/
/*                                */
/*************************************************************************/
void MakePS(PS **TempPS)
{
int done=0;
int allow_OK=0;
int time_OK=0;

PS *CheckListPS=NULL;

    (*TempPS) = (PS *)malloc(sizeof(PS));
    while(!done)
    {
        (*TempPS) -> x = (round(((float)rand()/RAND_MAX)*(xSize-1)));
        (*TempPS) -> y = (round(((float)rand()/RAND_MAX)*(ySize-1)));
#if !ALLOW_SOURCES_OUTSIDE
        if (((Area + ((*TempPS) -> x)*ySize + (*TempPS) -> y)->Type == TUMOUR)
           || ((Area + ((*TempPS) -> x)*ySize + (*TempPS) -> y)->Type == PERIPHERY))
            allow_OK = 1;

#else
        if (((Area + ((*TempPS) -> x)*ySize + (*TempPS) -> y)->Type == TUMOUR)
           || (((Area + ((*TempPS) -> x)*ySize + (*TempPS) -> y)->Type == MOVABLE_LOCATION))
           || (((Area + ((*TempPS) -> x)*ySize + (*TempPS) -> y)->Type == PERIPHERY)))
            allow_OK = 1;
#endif

#if !TIME_FACTOR /*then make sure that there are no other sources at this location*/
    /* run through linked list of sources and check them all */
        time_OK = 1;
        CheckListPS = FirstPS;/* the only list that matters in this list...*/
        while (CheckListPS)
        {
            if ((CheckListPS->x == (*TempPS)->x) && (CheckListPS->y == (*TempPS)->y))
            {
                //fprintf(stderr,"ON TOP OF OTHER SOURCE, NO GOOD\n");
                time_OK = 0;
            }
            CheckListPS = CheckListPS->NextPS;
        }
#else
        time_OK = 1;
#endif
        done = allow_OK&&time_OK;

        allow_OK = 0;
    }/*end while !done*/
```

```
    (*TempPS) -> DwellTime = DWELL_TIME;
    (*TempPS) -> NextPS = NULL;
}//end of MakePS
/*************************************************************************/


/*************************************************************************/
/*          Function : MovePS*/
/*                                 */
/*  Purpose : This function will move a specified PS*/
/*  Input : A PS.              */
/*  Output : Text to stderr.*/
/*                                 */
/*************************************************************************/
int MovePS(int WhichSource,PS *original_list,PS **new_list)
{
int SourceCounter=0;/* source to move */
int done=0;/* set to 1 if the source location is within the tumour */
PS *TempPS=NULL;/* For NewPS linked list */
PS *NewPS=NULL;/* For NewPS linked list */
PS *CurNewPS=NULL;/* For NewPS linked list */
PS *CheckListPS=NULL;/* for the check that the source has a unique location */
float TempRand=0.0f;/* To hold the random number generated */
int MultFactor=0;/* How far to move */
int missed_counter=0;/* this is how many times we have not been able to move the desired source */

    /*fprintf(stderr,"moving\n");*/
    if (original_list)
    {
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"In MoveSources and there are PointSources to move:\n");
    PrintPS(original_list);
    fprintf(stderr,"---\n");
#endif
    TempPS = original_list;
    }
    else
    {
/*Temp stuff so that it will work even if there are no sources*/
TempPS = original_list;

//      fprintf(stderr,"In MoveSources and there are no PointSources to move\n");
//      PrintPS(original_list);
//      fprintf(stderr,"---\n");
//      return(1);
    }
#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"Source number %i\n",WhichSource);
#endif


MultFactor=1;

    missed_counter = 0;
    SourceCounter = 0;
    while (TempPS)
    {
        NewPS = (PS *)malloc(sizeof(PS));
        if (SourceCounter == WhichSource)
        {
            done = 0;
            while(!done)
            {
            /* MOVE IN X DIRECTION */
                TempRand = (float)rand()/RAND_MAX;
#if DEBUG_MOVE_SOURCES
fprintf(stderr,"Rand # is: %f\n",TempRand);
#endif

                if ((TempRand < 0.333333f) && (TempPS -> x > MultFactor))
                    NewPS -> x = TempPS -> x - MultFactor;
                else if ((TempRand < 0.666666f) && (TempPS -> x < (xSize-1-MultFactor)))
```

```
                NewPS -> x = TempPS -> x + MultFactor;
            else
                NewPS -> x = TempPS -> x;/*TempPS -> x does not change*/

#if DEBUG_MOVE_SOURCES
fprintf(stderr,"Rand # is: %f\n",TempRand);
#endif

            /* MOVE IN Y DIRECTION */
                TempRand = (float)rand()/RAND_MAX;
                if ((TempRand < 0.333333f) && (TempPS -> y > MultFactor))
                    NewPS -> y = TempPS -> y - MultFactor;
                else if ((TempRand < 0.666666f) && (TempPS -> y < (ySize-1-MultFactor)))
                    NewPS -> y = TempPS -> y + MultFactor;
                else
                    NewPS -> y = TempPS -> y;/*TempPS -> y does not change*/

                done = 1;/* assume that we have made a good move */
#if !ALLOW_SOURCES_OUTSIDE
            if (((Area + ((NewPS) -> x)*ySize + (NewPS) -> y)->Type == TUMOUR)
               || ((Area + ((NewPS) -> x)*ySize + (NewPS) -> y)->Type == PERIPHERY))
                done = 1;
            else
            {
                missed_counter++;
                done = 0;
            }
#else/* make sure that point source has moved to a valid place */
            if (((Area + (ySize * (NewPS -> x)) +  NewPS -> y) -> Type == TUMOUR)
               || ((Area + ((NewPS) -> x)*ySize + (NewPS) -> y)->Type == PERIPHERY)
               || ((Area + (ySize * (NewPS -> x)) +  NewPS -> y) -> Type == MOVABLE_LOCATION))
                done = 1;
            else
            {
                missed_counter++;
                done = 0;
            }
#endif

#if !TIME_FACTOR/*then make sure that there are no other sources at this location*/
    /* run through linked list of sources and check them all */
            CheckListPS = original_list;/* the only list that matters in this list...*/
            while (CheckListPS && done)
            {
                if ((CheckListPS->x == NewPS->x) && (CheckListPS->y == NewPS->y))
                {
                    done = 0;
                    missed_counter++;
                }
                CheckListPS = CheckListPS->NextPS;
            }
#endif
            /* because only one source gets moved check to make sure that it did infact move*/
            if (done && (NewPS -> x == TempPS -> x) && (NewPS -> y == TempPS -> y))
                done = 0;


#if DEBUG_MOVE_SOURCES
    fprintf(stderr,"missed_counter:%d\n",missed_counter);
#endif

            if ((missed_counter == 24) && (!done))//then set it back to where it was
            {
                NewPS -> x = TempPS -> x;
                NewPS -> y = TempPS -> y;
                done = 1;
            }

            }/* end while not done */

            NewPS -> DwellTime = DWELL_TIME;
            NewPS -> NextPS = NULL;
```

```
        }/*end if SourceCounter == WhichSource*/
        else
        {
            NewPS -> x = TempPS -> x;
            NewPS -> y = TempPS -> y;
            NewPS -> DwellTime = TempPS -> DwellTime;
            NewPS -> NextPS = NULL;
        }/* end else */

        if (!(*new_list))
            (*new_list) = NewPS;
        else
            CurNewPS -> NextPS = NewPS;

        CurNewPS = NewPS;
        SourceCounter++;/* go to next source */
        TempPS = TempPS -> NextPS; /* go to next source */
    }/* end while TempPS */
    return(0);
}//end function MOvePS
/*************************************************************************/

/*************************************************************************/
/*          Function : NumPS*/
/*                                 */
/*  Purpose : This function will return the number of PS in a list*/
/*  Reason : To remove the number of sources as a global variable*/
/*  Input : A PS.                */
/*  Output : int which is the count of the number of sources*/
/*                                 */
/*************************************************************************/
int NumPS(PS *TheList)
{
    int counter=0;
    PS *TempPS=NULL;

    TempPS = TheList;
    while (TempPS)
    {
        counter++;
        TempPS = TempPS -> NextPS;
    }
    return(counter);
}// end function NumPS
/*************************************************************************/
```

E.40 RatioDlg1.cpp

```
// RatioDlg1.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "RatioDlg1.h"

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

/////////////////////////////////////////////////////////////////////////////
// CRatioDlg1 dialog


CRatioDlg1::CRatioDlg1(CWnd* pParent /*=NULL*/)
    : CDialog(CRatioDlg1::IDD, pParent)
{
```

```
    //{{AFX_DATA_INIT(CRatioDlg1)
    m_Ratio = 0.0f;
    //}}AFX_DATA_INIT
}


void CRatioDlg1::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CRatioDlg1)
    DDX_Text(pDX, IDC_RATIO, m_Ratio);
    DDV_MinMaxFloat(pDX, m_Ratio, 0.f, 1.f);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CRatioDlg1, CDialog)
    //{{AFX_MSG_MAP(CRatioDlg1)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRatioDlg1 message handlers
```

E.41 RatioDlg1.h

```
#if !defined(AFX_RATIODLG1_H__E9AAFED7_8243_11D3_807C_0090276F4183__INCLUDED_)
#define AFX_RATIODLG1_H__E9AAFED7_8243_11D3_807C_0090276F4183__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// RatioDlg1.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CRatioDlg1 dialog

class CRatioDlg1 : public CDialog
{
// Construction
public:
    CRatioDlg1(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CRatioDlg1)
    enum { IDD = IDD_RATIO };
    floatm_Ratio;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CRatioDlg1)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CRatioDlg1)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

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

#endif // !defined(AFX_RATIODLG1_H__E9AAFED7_8243_11D3_807C_0090276F4183__INCLUDED_)
```

E.42 Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Isocurve.rc
//
#define IDR_ISOCURTYPE_CNTR_IP          6
#define IDD_ABOUTBOX                    100
#define IDP_OLE_INIT_FAILED             100
#define IDP_FAILED_TO_CREATE            102
#define IDR_MAINFRAME                   128
#define IDR_ISOCURTYPE                  129
#define IDR_COOLINGTYPE                 130
#define IDD_ISOCURVEDLG                 131
#define IDD_DIALOG1                     132
#define IDD_VIEW                        132
#define IDR_CONTEXTMENU                 133
#define IDD_VIEWDATA                    134
#define IDB_SABIT                       135
#define IDD_ADDTYPE                     136
#define IDD_RATIO                       137
#define IDD_TRANSFORMS                  138
#define IDD_STATSREPORT                 140
#define IDD_NEWSHAPE                    141
#define IDD_ADDPOINTSOURCE              143
#define IDD_OUTPUTPROPS                 144
#define IDD_MULTFACT                    145
#define IDC_EDIT1                       1000
#define IDC_FILENAME                    1001
#define IDC_Y                           1001
#define IDC_HOMO                        1001
#define IDC_NUMINST                     1001
#define IDC_DOSE                        1002
#define IDC_PERCHECK                    1002
#define IDC_DWELLTIME                   1002
#define IDC_TYPE                        1003
#define IDC_FITNESS                     1003
#define IDC_X                           1004
#define IDC_POINT                       1005
#define IDC_TUMOR                       1006
#define IDC_PERIPHERY                   1007
#define IDC_OUTSIDE                     1008
#define IDC_DESIREDPS                   1009
#define IDC_RATIO                       1010
#define IDC_XT                          1011
#define IDC_TRANSLATIONS                1012
#define IDC_ROTATIONS                   1013
#define IDC_YT                          1014
#define IDC_HYP                         1014
#define IDC_ZT                          1015
#define IDC_CANVASX                     1015
#define IDC_XR                          1016
#define IDC_CANVASY                     1016
#define IDC_YR                          1017
#define IDC_RADIO1                      1017
#define IDC_ZR                          1018
#define IDC_RADIO2                      1018
#define IDC_SQUAREX                     1019
#define IDC_SQUAREY                     1020
#define IDC_RADIUS                      1021
#define IDC_CRCLPROP                    1022
#define IDC_SQRPROP                     1023
```

```
#define IDC_RAD                         1024
#define IDC_SQRX                        1025
#define IDC_SQRY                        1026
#define IDC_MANHATTAN                   1027
#define IDC_CONT                        1028
#define IDC_DISC                        1029
#define IDC_NEW                         1030
#define IDC_RADIO4                      1031
#define IDC_RADIO5                      1032
#define IDC_MULTFACT                    1035
#define ID_CANCEL_EDIT_CNTR             32768
#define ID_VIEW_ISOCURVE                32771
#define ID_VIEW_CLEARISOCURVE           32772
#define ID_CONTEXT_ADD                  32773
#define ID_CONTEXT_VIEW                 32774
#define ID_CONTEXT_DELETE               32775
#define ID_FILE_OUTPUT                  32776
#define ID_FILE_INPUTPREDICTED          32777
#define ID_VIEW_TRANSFORMS              32778
#define ID_VIEW_PERSPECTIVEORTHO        32779
#define ID_VIEW_RATIO                   32780
#define ID_VIEW_NEURALNETWORK           32781
#define ID_VIEW_VIEWPREDICTIONS         32782
#define ID_VIEW_COONSLINEAR             32783
#define ID_VIEW_STATSREPORT             32784
#define ID_FILE_NEWSHAPE                32785
#define ID_EDIT_ADDPOINTSOURCE          32786
#define ID_EDIT_RESTORESOURCEPOSITIONS  32787
#define ID_EDIT_STOREENERGYASDESIREDOUTPUT 32788
#define ID_VIEW_MULTFACTFORCEV          32789
#define ID_EDIT_CLEARALLPOINTSOURCES    32790
#define ID_ORTHOPERSPECT                32791
#define ID_SHOWHIDEISO                  32792
#define ID_PRESCRIBE                    32793
#define ID_VIEW_TOGGLEISOCURVE          32794

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        146
#define _APS_NEXT_COMMAND_VALUE         32795
#define _APS_NEXT_CONTROL_VALUE         1036
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

E.43 Speech.h

```
/****************************************************************************
Speech.H - Header file to use the Microsoft Speech APIs.

Copyright 1994 -1998 by Microsoft corporation.All rights reserved.
*/

#ifndef _SPEECH_
#define _SPEECH_

// Disable the warning for zero-length arrays in structures
#pragma warning(disable:4200)


/****************************************************************************
Defines common to all of the speech APIs.
*/

// Application  Speech API   Compiler Defines                    _S_UNICODE
```

```
// ------------------------------------------------------------------------------
//   ANSI        ANSI        <none>                                undefined
//   ANSI        Unicode     _S_UNICODE                            defined
//   Unicode     ANSI        (UNICODE || _UNICODE) && _S_ANSI      undefined
//   Unicode     Unicode     (UNICODE || _UNICODE)                 defined

#if (defined(UNICODE) || defined(_UNICODE)) && !defined(_S_ANSI)
#ifndef _S_UNICODE
#define _S_UNICODE
#endif
#endif

/*****************************************************************************
defines */
#define  SVFN_LEN     (262)
#define  LANG_LEN     (64)
#define  EI_TITLESIZE    (128)
#define  EI_DESCSIZE     (512)
#define  EI_FIXSIZE      (512)
#define  SVPI_MFGLEN     (64)
#define  SVPI_PRODLEN    (64)
#define  SVPI_COMPLEN    (64)
#define  SVPI_COPYRIGHTLEN (128)
#define  SVI_MFGLEN      (SVPI_MFGLEN)
#define  SETBIT(x)       ((DWORD)1 << (x))


// Error Macros
#define  FACILITY_SPEECH    (FACILITY_ITF)
#define  SPEECHERROR(x)     MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x200)
#define  AUDERROR(x)        MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x300)
#define  SRWARNING(x)       MAKE_SCODE(SEVERITY_SUCCESS, FACILITY_SPEECH, (x)+0x400)
#define  SRERROR(x)         MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x400)
#define  TTSERROR(x)        MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x500)
#define  VCMDERROR(x)       MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x600)
#define  VTXTERROR(x)       MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x700)
#define  LEXERROR(x)        MAKE_SCODE(SEVERITY_ERROR,   FACILITY_SPEECH, (x)+0x800)
#define  LEXWARNING(x)      MAKE_SCODE(SEVERITY_SUCCESS, FACILITY_SPEECH, (x)+0x800)

// Audio Errors
#define  AUDERR_NONE                      S_OK                        // 0x00000000
#define  AUDERR_BADDEVICEID               AUDERROR(1)                 // 0x80040301
#define  AUDERR_NEEDWAVEFORMAT            AUDERROR(2)                 // 0x80040302
#define  AUDERR_NOTSUPPORTED              E_NOTIMPL                   // 0x80004001
#define  AUDERR_NOTENOUGHDATA             SPEECHERROR(1)              // 0x80040201
#define  AUDERR_NOTPLAYING                AUDERROR(6)                 // 0x80040306
#define  AUDERR_INVALIDPARAM              E_INVALIDARG                // 0x80070057
#define  AUDERR_WAVEFORMATNOTSUPPORTED    SPEECHERROR(2)              // 0x80040202
#define  AUDERR_WAVEDEVICEBUSY            SPEECHERROR(3)              // 0x80040203
#define  AUDERR_WAVEDEVNOTSUPPORTED       AUDERROR(18)                // 0x80040312
#define  AUDERR_NOTRECORDING              AUDERROR(19)                // 0x80040313
#define  AUDERR_INVALIDFLAG               SPEECHERROR(4)              // 0x80040204
#define  AUDERR_INVALIDHANDLE             E_HANDLE                    // 0x80070006
#define  AUDERR_NODRIVER                  AUDERROR(23)                // 0x80040317
#define  AUDERR_HANDLEBUSY                AUDERROR(24)                // 0x80040318
#define  AUDERR_INVALIDNOTIFYSINK         AUDERROR(25)                // 0x80040319
#define  AUDERR_WAVENOTENABLED            AUDERROR(26)                // 0x8004031A
#define  AUDERR_ALREADYCLAIMED            AUDERROR(29)                // 0x8004031D
#define  AUDERR_NOTCLAIMED                AUDERROR(30)                // 0x8004031E
#define  AUDERR_STILLPLAYING              AUDERROR(31)                // 0x8004031F
#define  AUDERR_ALREADYSTARTED            AUDERROR(32)                // 0x80040320
#define  AUDERR_SYNCNOTALLOWED            AUDERROR(33)                // 0x80040321

// Speech Recognition Warnings
#define  SRWARN_BAD_LIST_PRONUNCIATION    SRWARNING(1)

// Speech Recognition Errors
#define  SRERR_NONE                       S_OK                        // 0x00000000
#define  SRERR_OUTOFDISK                  SPEECHERROR(5)              // 0x80040205
#define  SRERR_NOTSUPPORTED               E_NOTIMPL                   // 0x80004001
#define  SRERR_NOTENOUGHDATA              AUDERR_NOTENOUGHDATA        // 0x80040201
#define  SRERR_VALUEOUTOFRANGE            E_UNEXPECTED                // 0x8000FFFF
```

```
#define  SRERR_GRAMMARTOOCOMPLEX          SRERROR(6)                    // 0x80040406
#define  SRERR_GRAMMARWRONGTYPE           SRERROR(7)                    // 0x80040407
#define  SRERR_INVALIDWINDOW              OLE_E_INVALIDHWND             // 0x8004000F
#define  SRERR_INVALIDPARAM               E_INVALIDARG                  // 0x80070057
#define  SRERR_INVALIDMODE                SPEECHERROR(6)                // 0x80040206
#define  SRERR_TOOMANYGRAMMARS            SRERROR(11)                   // 0x8004040B
#define  SRERR_INVALIDLIST                SPEECHERROR(7)                // 0x80040207
#define  SRERR_WAVEDEVICEBUSY             AUDERR_WAVEDEVICEBUSY         // 0x80040203
#define  SRERR_WAVEFORMATNOTSUPPORTED     AUDERR_WAVEFORMATNOTSUPPORTED // 0x80040202
#define  SRERR_INVALIDCHAR                SPEECHERROR(8)                // 0x80040208
#define  SRERR_GRAMTOOCOMPLEX             SRERR_GRAMMARTOOCOMPLEX       // 0x80040406
#define  SRERR_GRAMTOOLARGE               SRERROR(17)                   // 0x80040411
#define  SRERR_INVALIDINTERFACE           E_NOINTERFACE                 // 0x80004002
#define  SRERR_INVALIDKEY                 SPEECHERROR(9)                // 0x80040209
#define  SRERR_INVALIDFLAG                AUDERR_INVALIDFLAG            // 0x80040204
#define  SRERR_GRAMMARERROR               SRERROR(22)                   // 0x80040416
#define  SRERR_INVALIDRULE                SRERROR(23)                   // 0x80040417
#define  SRERR_RULEALREADYACTIVE          SRERROR(24)                   // 0x80040418
#define  SRERR_RULENOTACTIVE              SRERROR(25)                   // 0x80040419
#define  SRERR_NOUSERSELECTED             SRERROR(26)                   // 0x8004041A
#define  SRERR_BAD_PRONUNCIATION          SRERROR(27)                   // 0x8004041B
#define  SRERR_DATAFILEERROR              SRERROR(28)                   // 0x8004041C
#define  SRERR_GRAMMARALREADYACTIVE       SRERROR(29)                   // 0x8004041D
#define  SRERR_GRAMMARNOTACTIVE           SRERROR(30)                   // 0x8004041E
#define  SRERR_GLOBALGRAMMARALREADYACTIVE SRERROR(31)                   // 0x8004041F
#define  SRERR_LANGUAGEMISMATCH           SRERROR(32)                   // 0x80040420
#define  SRERR_MULTIPLELANG               SRERROR(33)                   // 0x80040421
#define  SRERR_LDGRAMMARNOWORDS           SRERROR(34)                   // 0x80040422
#define  SRERR_NOLEXICON                  SRERROR(35)                   // 0x80040423
#define  SRERR_SPEAKEREXISTS              SRERROR(36)                   // 0x80040424
#define  SRERR_GRAMMARENGINEMISMATCH      SRERROR(37)                   // 0x80040425
#define  SRERR_BOOKMARKEXISTS             SRERROR(38)                   // 0x80040426
#define  SRERR_BOOKMARKDOESNOTEXIST       SRERROR(39)                   // 0x80040427
#define  SRERR_MICWIZARDCANCELED          SRERROR(40)                   // 0x80040428
#define  SRERR_WORDTOOLONG                SRERROR(41)                   // 0x80040429
#define  SRERR_BAD_WORD                   SRERROR(42)                   // 0x8004042A
#define  E_WRONGTYPE                      SPEECHERROR(13)               // 0x8004020C
#define  E_BUFFERTOOSMALL                 SPEECHERROR(14)               // 0x8004020D

// Text to Speech Errors
#define  TTSERR_NONE                      S_OK                          // 0x00000000
#define  TTSERR_INVALIDINTERFACE          E_NOINTERFACE                 // 0x80004002
#define  TTSERR_OUTOFDISK                 SRERR_OUTOFDISK               // 0x80040205
#define  TTSERR_NOTSUPPORTED              E_NOTIMPL                     // 0x80004001
#define  TTSERR_VALUEOUTOFRANGE           E_UNEXPECTED                  // 0x8000FFFF
#define  TTSERR_INVALIDWINDOW             OLE_E_INVALIDHWND             // 0x8004000F
#define  TTSERR_INVALIDPARAM              E_INVALIDARG                  // 0x80070057
#define  TTSERR_INVALIDMODE               SRERR_INVALIDMODE             // 0x80040206
#define  TTSERR_INVALIDKEY                SRERR_INVALIDKEY              // 0x80040209
#define  TTSERR_WAVEFORMATNOTSUPPORTED    AUDERR_WAVEFORMATNOTSUPPORTED // 0x80040202
#define  TTSERR_INVALIDCHAR               SRERR_INVALIDCHAR             // 0x80040208
#define  TTSERR_QUEUEFULL                 SPEECHERROR(10)               // 0x8004020A
#define  TTSERR_WAVEDEVICEBUSY            AUDERR_WAVEDEVICEBUSY         // 0x80040203
#define  TTSERR_NOTPAUSED                 TTSERROR(1)                   // 0x80040501
#define  TTSERR_ALREADYPAUSED             TTSERROR(2)                   // 0x80040502

// Voice Command Errors

/*
 *  Everything worked
 */
#define  VCMDERR_NONE                     S_OK                          // 0x00000000

/*
 *  Voice Commands could not allocate memory
 */
#define  VCMDERR_OUTOFMEM                 E_OUTOFMEMORY                 // 0x8007000E

/*
 *  Voice Commands could not store/retrieve a command set from the database
```

```
 */
#define  VCMDERR_OUTOFDISK               SRERR_OUTOFDISK              // 0x80040205

/*
 *  Function not implemented
 */
#define  VCMDERR_NOTSUPPORTED            E_NOTIMPL                    // 0x80004001

/*
 *  A parameter was passed that was out of the ranged of accepted values
 */
#define  VCMDERR_VALUEOUTOFRANGE         E_UNEXPECTED                 // 0x8000FFFF

/*
 *  A menu was too complex to compile a context-free grammar
 */
#define  VCMDERR_MENUTOOCOMPLEX          VCMDERROR(0x06)              //  0x80040606

/*
 *  Language mismatch between the speech recognition mode and menu trying
 *  to create
 */
#define  VCMDERR_MENUWRONGLANGUAGE       VCMDERROR(0x07)              // 0x80040607

/*
 *  An invalid window handle was passed to Voice Commands
 */
#define  VCMDERR_INVALIDWINDOW           OLE_E_INVALIDHWND            // 0x8004000F

/*
 *  Voice Commands detected a bad function parameter
 */
#define  VCMDERR_INVALIDPARAM            E_INVALIDARG                 // 0x80070057

/*
 *  This function cannot be completed right now, usually when trying to do
 *  some operation while no speech recognition site is established
 */
#define  VCMDERR_INVALIDMODE             SRERR_INVALIDMODE            // 0x80040206

/*
 *  There are too many Voice Commands menu
 */                                                                   // 0x8004060B
#define  VCMDERR_TOOMANYMENUS            VCMDERROR(0x0B)

/*
 *  Invalid list passed to ListSet/ListGet
 */
#define  VCMDERR_INVALIDLIST             SRERR_INVALIDLIST            // 0x80040207

/*
 *  Trying to open an existing menu that is not in the Voice Commands database
 */
#define  VCMDERR_MENUDOESNOTEXIST        VCMDERROR(0x0D)              // 0x8004060D

/*
 *  The function could not be completed because the menu is actively
 *  listening for commands
 */
#define  VCMDERR_MENUACTIVE              VCMDERROR(0x0E)              // 0x8004060E

/*
 *  No speech recognition engine is started
 */
#define  VCMDERR_NOENGINE                VCMDERROR(0x0F)              // 0x8004060F

/*
 *  Voice Commands could not acquire a Grammar interface from the speech
 *  recognition engine
 */
#define  VCMDERR_NOGRAMMARINTERFACE      VCMDERROR(0x10)              // 0x80040610
```

```
/*
 *  Voice Commands could not acquire a Find interface from the speech
 *  recognition engine
 */
#define  VCMDERR_NOFINDINTERFACE          VCMDERROR(0x11)              // 0x80040611

/*
 *  Voice Commands could not create a speech recognition enumerator
 */
#define  VCMDERR_CANTCREATESRENUM         VCMDERROR(0x12)              // 0x80040612

/*
 *  Voice Commands could get the appropriate site information to start a
 *  speech recognition engine
 */
#define  VCMDERR_NOSITEINFO               VCMDERROR(0x13)              // 0x80040613

/*
 *  Voice Commands could not find a speech recognition engine
 */
#define  VCMDERR_SRFINDFAILED             VCMDERROR(0x14)              // 0x80040614

/*
 *  Voice Commands could not create an audio source object
 */
#define  VCMDERR_CANTCREATEAUDIODEVICE    VCMDERROR(0x15)              // 0x80040615

/*
 *  Voice Commands could not set the appropriate device number in the
 *  audio source object
 */
#define  VCMDERR_CANTSETDEVICE            VCMDERROR(0x16)              // 0x80040616

/*
 *  Voice Commands could not select a speech recognition engine. Usually the
 *  error will occur when Voice Commands has enumerated and found an
 *  appropriate speech recognition engine, then it is not able to actually
 *  select/start the engine. There are different reasons that the engine won't
 *  start, but the most common is that there is no wave in device.
 */
#define  VCMDERR_CANTSELECTENGINE         VCMDERROR(0x17)              // 0x80040617

/*
 *  Voice Commands could not create a notfication sink for engine
 *  notifications
 */
#define  VCMDERR_CANTCREATENOTIFY         VCMDERROR(0x18)              // 0x80040618

/*
 *  Voice Commands could not create internal data structures.
 */
#define  VCMDERR_CANTCREATEDATASTRUCTURES VCMDERROR(0x19)              // 0x80040619

/*
 *  Voice Commands could not initialize internal data structures
 */
#define  VCMDERR_CANTINITDATASTRUCTURES   VCMDERROR(0x1A)              // 0x8004061A

/*
 *  The menu does not have an entry in the Voice Commands cache
 */
#define  VCMDERR_NOCACHEDATA              VCMDERROR(0x1B)              // 0x8004061B

/*
 *  The menu does not have commands
 */
#define  VCMDERR_NOCOMMANDS               VCMDERROR(0x1C)              // 0x8004061C

/*
 *  Voice Commands cannot extract unique words needed for the engine grammar
 */
#define  VCMDERR_CANTXTRACTWORDS          VCMDERROR(0x1D)              // 0x8004061D
```

```
/*
 *  Voice Commands could not get the command set database name
 */
#define  VCMDERR_CANTGETDBNAME            VCMDERROR(0x1E)              // 0x8004061E

/*
 *  Voice Commands could not create a registry key
 */
#define  VCMDERR_CANTCREATEKEY            VCMDERROR(0x1F)              // 0x8004061F

/*
 *  Voice Commands could not create a new database name
 */
#define  VCMDERR_CANTCREATEDBNAME         VCMDERROR(0x20)              // 0x80040620

/*
 *  Voice Commands could not update the registry
 */
#define  VCMDERR_CANTUPDATEREGISTRY       VCMDERROR(0x21)              // 0x80040621

/*
 *  Voice Commands could not open the registry
 */
#define  VCMDERR_CANTOPENREGISTRY         VCMDERROR(0x22)              // 0x80040622

/*
 *  Voice Commands could not open the command set database
 */
#define  VCMDERR_CANTOPENDATABASE         VCMDERROR(0x23)              // 0x80040623

/*
 *  Voice Commands could not create a database storage object
 */
#define  VCMDERR_CANTCREATESTORAGE        VCMDERROR(0x24)              // 0x80040624

/*
 *  Voice Commands could not do CmdMimic
 */
#define  VCMDERR_CANNOTMIMIC              VCMDERROR(0x25)              // 0x80040625

/*
 *  A menu of this name already exist
 */
#define  VCMDERR_MENUEXIST                VCMDERROR(0x26)              // 0x80040626

/*
 *  A menu of this name is open and cannot be deleted right now
 */
#define  VCMDERR_MENUOPEN                 VCMDERROR(0x27)              // 0x80040627


// Voice Text Errors
#define  VTXTERR_NONE                     S_OK                         // 0x00000000

/*
 *  Voice Text failed to allocate memory it needed
 */
#define  VTXTERR_OUTOFMEM                 E_OUTOFMEMORY                // 0x8007000E

/*
 *  An empty string ("") was passed to the Speak function
 */
#define  VTXTERR_EMPTYSPEAKSTRING         SPEECHERROR(0x0b)            // 0x8004020B

/*
 *  An invalid parameter was passed to a Voice Text function
 */
#define  VTXTERR_INVALIDPARAM             E_INVALIDARG                 // 0x80070057

/*
 *  The called function cannot be done at this time. This usually occurs
```

```
 *  when trying to call a function that needs a site, but no site has been
 *  registered.
 */
#define  VTXTERR_INVALIDMODE              SRERR_INVALIDMODE             // 0x80040206

/*
 *  No text-to-speech engine is started
 */
#define  VTXTERR_NOENGINE                 VTXTERROR(0x0F)               // 0x8004070F

/*
 *  Voice Text could not acquire a Find interface from the text-to-speech
 *  engine
 */
#define  VTXTERR_NOFINDINTERFACE          VTXTERROR(0x11)               // 0x80040711

/*
 *  Voice Text could not create a text-to-speech enumerator
 */
#define  VTXTERR_CANTCREATETTSENUM        VTXTERROR(0x12)               // 0x80040712

/*
 *  Voice Text could get the appropriate site information to start a
 *  text-to-speech engine
 */
#define  VTXTERR_NOSITEINFO               VTXTERROR(0x13)               // 0x80040713

/*
 *  Voice Text could not find a text-to-speech engine
 */
#define  VTXTERR_TTSFINDFAILED            VTXTERROR(0x14)               // 0x80040714

/*
 *  Voice Text could not create an audio destination object
 */
#define  VTXTERR_CANTCREATEAUDIODEVICE    VTXTERROR(0x15)               // 0x80040715

/*
 *  Voice Text could not set the appropriate device number in the
 *  audio destination object
 */
#define  VTXTERR_CANTSETDEVICE            VTXTERROR(0x16)               // 0x80040716

/*
 *  Voice Text could not select a text-to-speech engine. Usually the
 *  error will occur when Voice Text has enumerated and found an
 *  appropriate text-to-speech engine, then it is not able to actually
 *  select/start the engine.
 */
#define  VTXTERR_CANTSELECTENGINE         VTXTERROR(0x17)               // 0x80040717

/*
 *  Voice Text could not create a notfication sink for engine
 *  notifications
 */
#define  VTXTERR_CANTCREATENOTIFY         VTXTERROR(0x18)               // 0x80040718

/*
 *  Voice Text is disabled at this time
 */
#define  VTXTERR_NOTENABLED               VTXTERROR(0x19)               // 0x80040719

#define  VTXTERR_OUTOFDISK                SRERR_OUTOFDISK               // 0x80040205
#define  VTXTERR_NOTSUPPORTED             E_NOTIMPL                     // 0x80004001
#define  VTXTERR_NOTENOUGHDATA            AUDERR_NOTENOUGHDATA          // 0x80040201
#define  VTXTERR_QUEUEFULL                TTSERR_QUEUEFULL              // 0x8004020A
#define  VTXTERR_VALUEOUTOFRANGE          E_UNEXPECTED                  // 0x8000FFFF
#define  VTXTERR_INVALIDWINDOW            OLE_E_INVALIDHWND             // 0x8004000F
#define  VTXTERR_WAVEDEVICEBUSY           AUDERR_WAVEDEVICEBUSY         // 0x80040203
#define  VTXTERR_WAVEFORMATNOTSUPPORTED   AUDERR_WAVEFORMATNOTSUPPORTED // 0x80040202
#define  VTXTERR_INVALIDCHAR              SRERR_INVALIDCHAR             // 0x80040208
```

```
// ILexPronounce errors
#define  LEXERR_INVALIDTEXTCHAR          LEXERROR(0x01)              // 0x80040801
#define  LEXERR_INVALIDSENSE             LEXERROR(0x02)              // 0x80040802
#define  LEXERR_NOTINLEX                 LEXERROR(0x03)              // 0x80040803
#define  LEXERR_OUTOFDISK                LEXERROR(0x04)              // 0x80040804
#define  LEXERR_INVALIDPRONCHAR          LEXERROR(0x05)              // 0x80040805
#define  LEXERR_ALREADYINLEX             LEXWARNING(0x06)            // 0x00040806
#define  LEXERR_PRNBUFTOOSMALL           LEXERROR(0x07)              // 0x80040807
#define  LEXERR_ENGBUFTOOSMALL           LEXERROR(0x08)              // 0x80040808
#define  LEXERR_INVALIDLEX               LEXERROR(0x09)              // 0x80040809


/**************************************************************************
typedefs */

typedef LPUNKNOWN FAR * PIUNKNOWN;

typedef struct {
   PVOID    pData;
   DWORD    dwSize;
   } SDATA, * PSDATA;



typedef struct {
   LANGID   LanguageID;
   WCHAR    szDialect[LANG_LEN];
   } LANGUAGEW, FAR * PLANGUAGEW;

typedef struct {
   LANGID   LanguageID;
   CHAR     szDialect[LANG_LEN];
   } LANGUAGEA, FAR * PLANGUAGEA;

#ifdef  _S_UNICODE
#define LANGUAGE    LANGUAGEW
#define PLANGUAGE   PLANGUAGEW
#else
#define LANGUAGE    LANGUAGEA
#define PLANGUAGE   PLANGUAGEA
#endif  // _S_UNICODE


typedef struct {
   HRESULT  hRes;
   WCHAR    szStrings[512];
   } SPCHERRORW, FAR * PSPCHERRORW;

typedef struct {
   HRESULT  hRes;
   CHAR     szStrings[512];
   } SPCHERRORA, FAR * PSPCHERRORA;

#ifdef  _S_UNICODE
#define SPCHERROR    SPCHERRORW
#define PSPCHERROR   PSPCHERRORW
#else
#define SPCHERROR    SPCHERRORA
#define PSPCHERROR   PSPCHERRORA
#endif  // _S_UNICODE


typedef unsigned __int64 QWORD, * PQWORD;

typedef enum {
   CHARSET_TEXT            = 0,
   CHARSET_IPAPHONETIC     = 1,
   CHARSET_ENGINEPHONETIC = 2
   } VOICECHARSET;
```

```
typedef enum _VOICEPARTOFSPEECH {
   VPS_UNKNOWN = 0,
   VPS_NOUN = 1,
   VPS_VERB = 2,
   VPS_ADVERB = 3,
   VPS_ADJECTIVE = 4,
   VPS_PROPERNOUN = 5,
   VPS_PRONOUN = 6,
   VPS_CONJUNCTION = 7,
   VPS_CARDINAL = 8,
   VPS_ORDINAL = 9,
   VPS_DETERMINER = 10,
   VPS_QUANTIFIER = 11,
   VPS_PUNCTUATION = 12,
   VPS_CONTRACTION = 13,
   VPS_INTERJECTION = 14,
   VPS_ABBREVIATION = 15,
   VPS_PREPOSITION = 16
   } VOICEPARTOFSPEECH;


typedef struct {
   DWORD   dwNextPhonemeNode;
   DWORD   dwUpAlternatePhonemeNode;
   DWORD   dwDownAlternatePhonemeNode;
   DWORD   dwPreviousPhonemeNode;
   DWORD   dwWordNode;
   QWORD   qwStartTime;
   QWORD   qwEndTime;
   DWORD   dwPhonemeScore;
   WORD    wVolume;
   WORD    wPitch;
   } SRRESPHONEMENODE, *PSRRESPHONEMENODE;


typedef struct {
   DWORD   dwNextWordNode;
   DWORD   dwUpAlternateWordNode;
   DWORD   dwDownAlternateWordNode;
   DWORD   dwPreviousWordNode;
   DWORD   dwPhonemeNode;
   QWORD   qwStartTime;
   QWORD   qwEndTime;
   DWORD   dwWordScore;
   WORD      wVolume;
   WORD      wPitch;
   VOICEPARTOFSPEECH   pos;
   DWORD   dwCFGParse;
   DWORD   dwCue;
   } SRRESWORDNODE, * PSRRESWORDNODE;


/*****************************************************************************
interfaces */

/*
 * IAttributes
 */

/* attribute values */
#define  TTSBASEATTR             0x1000
#define  SRBASEATTR              0x2000
#define  VDCTBASEATTR            0x3000
#define  VCMDBASEATTR            0x4000
#define  VTXTBASEATTR            0x5000
#define  AUDBASEATTR             0x6000

#define  TTSATTR_PITCH           (1)
#define  TTSATTR_REALTIME        (0)
#define  TTSATTR_SPEED           (2)
#define  TTSATTR_VOLUME          (3)
```

```
#define  TTSATTR_PITCHRANGE      (TTSBASEATTR + 5)
#define  TTSATTR_PITCHRANGEDEFAULT (TTSBASEATTR + 6)
#define  TTSATTR_PITCHRANGEMAX   (TTSBASEATTR + 7)
#define  TTSATTR_PITCHRANGEMIN   (TTSBASEATTR + 8)
#define  TTSATTR_PITCHRANGERELATIVE (TTSBASEATTR + 9)
#define  TTSATTR_PITCHRANGERELATIVEMAX (TTSBASEATTR + 10)
#define  TTSATTR_PITCHRANGERELATIVEMIN (TTSBASEATTR + 11)
#define  TTSATTR_PITCHRELATIVE   (TTSBASEATTR + 12)
#define  TTSATTR_PITCHRELATIVEMAX (TTSBASEATTR + 13)
#define  TTSATTR_PITCHRELATIVEMIN (TTSBASEATTR + 14)
#define  TTSATTR_PITCHDEFAULT    (TTSBASEATTR + 15)
#define  TTSATTR_PITCHMAX        (TTSBASEATTR + 16)
#define  TTSATTR_PITCHMIN        (TTSBASEATTR + 17)
#define  TTSATTR_SPEEDRELATIVE   (TTSBASEATTR + 18)
#define  TTSATTR_SPEEDRELATIVEMAX (TTSBASEATTR + 19)
#define  TTSATTR_SPEEDRELATIVEMIN (TTSBASEATTR + 20)
#define  TTSATTR_SPEEDDEFAULT    (TTSBASEATTR + 21)
#define  TTSATTR_SPEEDMAX        (TTSBASEATTR + 22)
#define  TTSATTR_SPEEDMIN        (TTSBASEATTR + 23)
#define  TTSATTR_THREADPRIORITY  (TTSBASEATTR + 24)
#define  TTSATTR_SINKFLAGS       (TTSBASEATTR + 25)
#define  TTSATTR_VOLUMEDEFAULT   (TTSBASEATTR + 26)

#define  SRATTR_AUTOGAIN         (1)
#define  SRATTR_ECHO             (3)
#define  SRATTR_ENERGYFLOOR      (4)
#define  SRATTR_MICROPHONE       (5)
#define  SRATTR_REALTIME         (6)
#define  SRATTR_SPEAKER          (7)
#define  SRATTR_TIMEOUT_COMPLETE (8)
#define  SRATTR_TIMEOUT_INCOMPLETE (SRBASEATTR + 8)
#define  SRATTR_THRESHOLD        (2)
#define  SRATTR_ACCURACYSLIDER   (SRBASEATTR + 10)
#define  SRATTR_LEVEL            (SRBASEATTR + 11)
#define  SRATTR_LISTENINGSTATE   (SRBASEATTR + 12)
#define  SRATTR_RESULTSINFO      (SRBASEATTR + 13)
#define  SRATTR_RESULTSINFO_POSSIBLE (SRBASEATTR + 14)
#define  SRATTR_SINKFLAGS        (SRBASEATTR + 15)
#define  SRATTR_THREADPRIORITY   (SRBASEATTR + 16)

#define  VDCTATTR_AWAKESTATE     (VDCTBASEATTR + 1)
#define  VDCTATTR_MODE           (VDCTBASEATTR + 2)
#define  VDCTATTR_MEMORY         (VDCTBASEATTR + 3)
#define  VDCTATTR_CORRECTIONRECT (VDCTBASEATTR + 4)
#define  VDCTATTR_VISIBLETEXTSTART (VDCTBASEATTR + 5)
#define  VDCTATTR_VISIBLETEXTCHARS (VDCTBASEATTR + 6)
#define  VDCTATTR_INSERTIONGUI   (VDCTBASEATTR + 7)   // turn on ISRGramInsertionGUI

#define  VCMDATTR_AWAKESTATE     (VCMDBASEATTR + 1)
#define  VCMDATTR_DEVICE         (VCMDBASEATTR + 2)
#define  VCMDATTR_ENABLED        (VCMDBASEATTR + 3)
#define  VCMDATTR_SRMODE         (VCMDBASEATTR + 4)

#define  AUDATTR_USELOWPRIORITY  (AUDBASEATTR + 1) // MMAudioSource: if true, send low priority message
to card when starting
#define  AUDATTR_AUTORETRY       (AUDBASEATTR + 2) // MMAudioSource: if true, keep retrying to open wave
device if can't initially
#define  AUDATTR_TIMERMS         (AUDBASEATTR + 3) // MMAudioDest: milliseconds between timer ticks, when
send out buffers, etc.
#define  AUDATTR_DIRECTFLAGS     (AUDBASEATTR + 4) // DirectSoundAudioDest: Flages to send into Create-
SoundBuffer
                                                   // set these values before waveformatset. Use to do
3-d tts

// SRATTR_RESULTSINFO flags
#define  SRRI_AUDIO SETBIT(0)
#define  SRRI_AUDIO_UNCOMPRESSED SETBIT(1)
#define  SRRI_ALTERNATIVES       SETBIT(2)
#define  SRRI_WORDGRAPH          SETBIT(3)
#define  SRRI_PHONEMEGRAPH       SETBIT(4)

// SRATTR_SINKFLAGS flags
```

```
#define  SRASF_ATTRIBUTES         SETBIT(0)
#define  SRASF_INTERFERENCE       SETBIT(1)
#define  SRASF_SOUND              SETBIT(2)
#define  SRASF_UTTERANCEBEGIN     SETBIT(3)
#define  SRASF_UTTERANCEEND       SETBIT(4)
#define  SRASF_VUMETER            SETBIT(5)
#define  SRASF_PHRASEHYPOTHESIS   SETBIT(6)
#define  SRASF_TRAINING           SETBIT(7)
#define  SRASF_ERRORWARNING       SETBIT(8)

#undef   INTERFACE
#define  INTERFACE   IAttributesW

// {1D59DED1-E367-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_IAttributesW,
0x1d59ded1, 0xe367, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IAttributesW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // AttributesW members
   STDMETHOD (DWORDGet)       (DWORD, DWORD*) PURE;
   STDMETHOD (DWORDSet)       (DWORD, DWORD) PURE;
   STDMETHOD (StringGet)      (DWORD, PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (StringSet)      (DWORD, PCWSTR) PURE;
   STDMETHOD (MemoryGet)      (DWORD, PVOID*, DWORD*) PURE;
   STDMETHOD (MemorySet)      (DWORD, PVOID, DWORD) PURE;
   };

typedef IAttributesW FAR * PIATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   IAttributesA

// {1D59DED2-E367-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_IAttributesA,
0x1d59ded2, 0xe367, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IAttributesA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // AttributesA members
   STDMETHOD (DWORDGet)       (DWORD, DWORD*) PURE;
   STDMETHOD (DWORDSet)       (DWORD, DWORD) PURE;
   STDMETHOD (StringGet)      (DWORD, PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (StringSet)      (DWORD, PCSTR) PURE;
   STDMETHOD (MemoryGet)      (DWORD, PVOID*, DWORD*) PURE;
   STDMETHOD (MemorySet)      (DWORD, PVOID, DWORD) PURE;
   };

typedef IAttributesA FAR * PIATTRIBUTESA;


#ifdef _S_UNICODE
 #define IAttributes        IAttributesW
 #define IID_IAttributes    IID_IAttributesW
 #define PIATTRIBUTES       PIATTRIBUTESW

#else
 #define IAttributes        IAttributesA
 #define IID_IAttributes    IID_IAttributesA
 #define PIATTRIBUTES       PIATTRIBUTESA
```

```
#endif   // _S_UNICODE


/*
 * ISpchError
 */

#undef   INTERFACE
#define  INTERFACE   ISpchErrorW

// {9B445335-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISpchErrorW,
0x9b445335, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISpchErrorW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // SpchError members
   STDMETHOD (LastErrorGet) (THIS_ SPCHERRORW *) PURE;
   STDMETHOD (ErrorMessageGet) (THIS_ WCHAR *, DWORD, DWORD *) PURE;
   };

typedef ISpchErrorW FAR * PISPCHERRORW;


#undef   INTERFACE
#define  INTERFACE   ISpchErrorA

// {9B445336-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISpchErrorA,
0x9b445336, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISpchErrorA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // SpchError members
   STDMETHOD (LastErrorGet) (THIS_ SPCHERRORA *) PURE;
   STDMETHOD (ErrorMessageGet) (THIS_ CHAR *, DWORD, DWORD *) PURE;
   };

typedef ISpchErrorA FAR * PISPCHERRORA;
#ifdef _S_UNICODE
 #define ISpchError        ISpchErrorW
 #define IID_ISpchError    IID_ISpchErrorW
 #define PISPCHERROR       PISPCHERRORW

#else
 #define ISpchError        ISpchErrorA
 #define IID_ISpchError    IID_ISpchErrorA
 #define PISPCHERROR       PISPCHERRORA

#endif   // _S_UNICODE




/*
 * ILexPronounce
 */

#undef   INTERFACE
#define  INTERFACE   ILexPronounceW
```

```
DEFINE_GUID(IID_ILexPronounceW, 0x090CD9A2, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ILexPronounceW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // LexPronounceW members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCWSTR, PCWSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCWSTR, WORD, PWSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCWSTR, WORD) PURE;
   };

typedef ILexPronounceW FAR * PILEXPRONOUNCEW;


#undef   INTERFACE
#define  INTERFACE   ILexPronounceA

DEFINE_GUID(IID_ILexPronounceA, 0x2F26B9C0L, 0xDB31, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ILexPronounceA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // LexPronounceA members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCSTR, PCSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCSTR, WORD, PSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCSTR, WORD) PURE;
   };

typedef ILexPronounceA FAR * PILEXPRONOUNCEA;


#ifdef _S_UNICODE
 #define ILexPronounce         ILexPronounceW
 #define IID_ILexPronounce     IID_ILexPronounceW
 #define PILEXPRONOUNCE        PILEXPRONOUNCEW

#else
 #define ILexPronounce         ILexPronounceA
 #define IID_ILexPronounce     IID_ILexPronounceA
 #define PILEXPRONOUNCE        PILEXPRONOUNCEA

#endif   // _S_UNICODE


/*
 * ILexPronounce2
 */

#define  ILP2_ACTIVE           SETBIT(0)
#define  ILP2_USER             SETBIT(1)
#define  ILP2_BACKUP           SETBIT(2)
#define  ILP2_LTS              SETBIT(3)

#undef   INTERFACE
#define  INTERFACE   ILexPronounce2W
```

```
// {D6E56341-B3EA-11d1-AFA5-0000F81E880D}
DEFINE_GUID(IID_ILexPronounce2W, 0xd6e56341, 0xb3ea, 0x11d1, 0xaf, 0xa5, 0x0, 0x0, 0xf8, 0x1e, 0x88,
0xd);

DECLARE_INTERFACE_ (ILexPronounce2W, ILexPronounceW) {

   // LexPronounceW members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCWSTR, PCWSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCWSTR, WORD, PWSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCWSTR, WORD) PURE;

   // LexPronounce2W members
   STDMETHOD (AddTo)            (THIS_ DWORD, VOICECHARSET, PCWSTR, PCWSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (GetFrom)            (THIS_ DWORD, VOICECHARSET, PCWSTR, WORD, PWSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (RemoveFrom)         (THIS_ DWORD, PCWSTR, WORD) PURE;
   STDMETHOD (QueryLexicons)  (THIS_ BOOL, DWORD *) PURE;
   STDMETHOD (ChangeSpelling) (THIS_ DWORD, PCWSTR, PCWSTR) PURE;
   };

typedef ILexPronounce2W FAR * PILEXPRONOUNCE2W;


#undef   INTERFACE
#define  INTERFACE   ILexPronounce2A

// {D6E56342-B3EA-11d1-AFA5-0000F81E880D}
DEFINE_GUID(IID_ILexPronounce2A, 0xd6e56342, 0xb3ea, 0x11d1, 0xaf, 0xa5, 0x0, 0x0, 0xf8, 0x1e, 0x88,
0xd);

DECLARE_INTERFACE_ (ILexPronounce2A, ILexPronounceA) {

   // LexPronounceA members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCSTR, PCSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCSTR, WORD, PSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCSTR, WORD) PURE;

   // LexPronounceA members
   STDMETHOD (AddTo)            (THIS_ DWORD, VOICECHARSET, PCSTR, PCSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD) PURE;
   STDMETHOD (GetFrom)            (THIS_ DWORD, VOICECHARSET, PCSTR, WORD, PSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (RemoveFrom)         (THIS_ DWORD, PCSTR, WORD) PURE;
   STDMETHOD (QueryLexicons)  (THIS_ BOOL, DWORD *) PURE;
   STDMETHOD (ChangeSpelling) (THIS_ DWORD, PCSTR, PCSTR) PURE;
   };

typedef ILexPronounce2A FAR * PILEXPRONOUNCE2A;


#ifdef _S_UNICODE
 #define ILexPronounce2        ILexPronounce2W
 #define IID_ILexPronounce2    IID_ILexPronounce2W
 #define PILEXPRONOUNCE2       PILEXPRONOUNCE2W

#else
 #define ILexPronounce2        ILexPronounce2A
 #define IID_ILexPronounce2    IID_ILexPronounce2A
 #define PILEXPRONOUNCE2       PILEXPRONOUNCE2A

#endif   // _S_UNICODE

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

```
Speech Tools
*/

/* defines */
// ISTMicWizard->Wizard
#define STMWU_CNC           0
#define STMWU_DICTATION     1
#define STMWU_LOWERGAIN     0x10000     // if on, tries for a lower gain
#define STMWU_NOAUTOGAIN    0x20000     // if on, doesn't adjust the autogain bit

#define STMWF_CANSKIP       SETBIT(0)

// MICWIZARDINFO
#define STMWI_UNKNOWN       0
#define STMWI_CLOSETALK     1
#define STMWI_EARPIECE      2
#define STMWI_HANDSET       3
#define STMWI_CLIPON        4
#define STMWI_DESKTOP       5
#define STMWI_HANDHELD      6
#define STMWI_TOPMONITOR    7
#define STMWI_INMONITOR     8
#define STMWI_KEYBOARD      9
#define STMWI_REMOTE        10

#define STMWIS_UNKNOWN      0
#define STMWIS_SPEAKERS     1
#define STMWIS_HEADPHONES   2
#define STMWIS_BOTH         3

/* structures */

typedef struct {
   BOOL     fHasRunMicWizard;
   DWORD    dwWaveInDevice;
   DWORD    dwMicrophoneType;
   WCHAR    szMicString[64];
} MICWIZARDINFO, *PMICWIZARDINFO;

typedef struct {
   BOOL     fHasRunMicWizard;
   DWORD    dwWaveInDevice;
   DWORD    dwMicrophoneType;
   WCHAR    szMicString[64];
   DWORD    dwSpeakers;
   DWORD    dwLastMicLevel;
   DWORD    dwMicLevelDeviceID;
} MICWIZARDINFO2, *PMICWIZARDINFO2;

// STLexDlg
#define  STLD_DISABLEREMOVE    SETBIT(0)
#define  STLD_DISABLEADD       SETBIT(1)
#define  STLD_FORCEEDIT        SETBIT(2)
#define  STLD_DISABLEPRONADDREMOVE SETBIT(3)
#define  STLD_TEST             SETBIT(4)
#define  STLD_DISABLERENAME    SETBIT(5)
#define  STLD_CHANGEPRONADDS   SETBIT(6)

/* class IDs */
// {B9F11A95-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(CLSID_STMicWizard,
0xb9f11a95, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {92655FB1-ADF9-11d1-BEB9-006008317CE8}
DEFINE_GUID(CLSID_STLexDlg,
0x92655fb1, 0xadf9, 0x11d1, 0xbe, 0xb9, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

// {B9F11A96-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(CLSID_STPhoneConv,
0xb9f11a96, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);
```

```
/* interfaces */
// ISTMicWizard Interface

#undef   INTERFACE
#define  INTERFACE   ISTMicWizard

// {B9F11A97-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(IID_ISTMicWizard,
0xb9f11a97, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (ISTMicWizard, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISTMicWizard members
   STDMETHOD (InfoGet) (THIS_ PMICWIZARDINFO) PURE;
   STDMETHOD (InfoSet) (THIS_ PMICWIZARDINFO) PURE;
   STDMETHOD (Wizard) (THIS_ HWND, DWORD, DWORD, DWORD, DWORD) PURE;
   };

typedef ISTMicWizard FAR * PISTMICWIZARD;


// ISTLexDlg Interface

#undef   INTERFACE
#define  INTERFACE   ISTLexDlg

// {A27B8A01-ADF9-11d1-BEB9-006008317CE8}
DEFINE_GUID(IID_ISTLexDlg,
0xa27b8a01, 0xadf9, 0x11d1, 0xbe, 0xb9, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISTLexDlg, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISTLexDlg members
   STDMETHOD (LexDlg) (THIS_ HWND, PCWSTR, DWORD, LANGID,
      LPUNKNOWN, LPUNKNOWN, LPUNKNOWN, PCWSTR, PCWSTR) PURE;
   };

typedef ISTLexDlg FAR * PISTLEXDLG;

// ISTPhoneConv Interface

#undef   INTERFACE
#define  INTERFACE   ISTPhoneConv

// {B9F11A98-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(IID_ISTPhoneConv,
0xb9f11a98, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (ISTPhoneConv, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISTPhoneConv members
   STDMETHOD (FromIPA) (THIS_ LPCWSTR, DWORD *, LPWSTR, DWORD, DWORD*) PURE;
   STDMETHOD (Get) (THIS_ SDATA *) PURE;
   STDMETHOD (SetByLang) (THIS_ LANGID) PURE;
   STDMETHOD (SetByText) (THIS_ LPCWSTR) PURE;
   STDMETHOD (ToIPA) (THIS_ LPCWSTR, DWORD *, LPWSTR, DWORD, DWORD*) PURE;
   };

typedef ISTPhoneConv FAR * PISTPHONECONV;
```

```
/*****************************************************************************
Audio source/destiantion API
*/

/*****************************************************************************
defines */

// AudioStop
#define      IANSRSN_NODATA             0
#define      IANSRSN_PRIORITY           1
#define      IANSRSN_INACTIVE           2
#define      IANSRSN_EOF                3

// IAudioSourceInstrumented::StateSet
#define          IASISTATE_PASSTHROUGH      0
#define          IASISTATE_PASSNOTHING      1
#define          IASISTATE_PASSREADFROMWAVE 2
#define          IASISTATE_PASSWRITETOWAVE  3

/*****************************************************************************
typedefs */

/*****************************************************************************
Class IDs */
// {CB96B400-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(CLSID_MMAudioDest,
0xcb96b400, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

// {D24FE500-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(CLSID_MMAudioSource,
0xd24fe500, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

// {D4023720-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(CLSID_InstAudioSource,
0xd4023720, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {D4123720-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(CLSID_SuspAudioSource,
0xd4123720, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {D4523720-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(CLSID_AudioSourceFile,
0xd4523720, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {D4523720-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(CLSID_AudioDestFile,
0xd4623720, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {B9F11A90-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(CLSID_AudioDestDirect,
0xb9f11a90, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

// {32C35401-D04F-11d0-99B3-00AA004CD65C}
DEFINE_GUID(CLSID_AudioSourceDirect,
0x32c35401, 0xd04f, 0x11d0, 0x99, 0xb3, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);


/*****************************************************************************
interfaces */

// IAudio
#undef   INTERFACE
#define  INTERFACE   IAudio

// {F546B340-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudio,
0xf546b340, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudio, IUnknown) {
   // IUnknown members
```

```
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IAudio members
    STDMETHOD (Flush)           (THIS) PURE;
    STDMETHOD (LevelGet)        (THIS_ DWORD *) PURE;
    STDMETHOD (LevelSet)        (THIS_ DWORD) PURE;
    STDMETHOD (PassNotify)      (THIS_ PVOID, IID) PURE;
    STDMETHOD (PosnGet)         (THIS_ PQWORD) PURE;
    STDMETHOD (Claim)           (THIS) PURE;
    STDMETHOD (UnClaim)         (THIS) PURE;
    STDMETHOD (Start)           (THIS) PURE;
    STDMETHOD (Stop)            (THIS) PURE;
    STDMETHOD (TotalGet)        (THIS_ PQWORD) PURE;
    STDMETHOD (ToFileTime)      (THIS_ PQWORD, FILETIME *) PURE;
    STDMETHOD (WaveFormatGet)   (THIS_ PSDATA) PURE;
    STDMETHOD (WaveFormatSet)   (THIS_ SDATA) PURE;
    };

typedef IAudio FAR * PIAUDIO;

// IAudioDest

#undef   INTERFACE
#define  INTERFACE   IAudioDest

// {2EC34DA0-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudioDest,
0x2ec34da0, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudioDest, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IAudioDest members
    STDMETHOD (FreeSpace)       (THIS_ DWORD *, BOOL *) PURE;
    STDMETHOD (DataSet)         (THIS_ PVOID, DWORD) PURE;
    STDMETHOD (BookMark)        (THIS_ DWORD) PURE;
    };

typedef IAudioDest FAR * PIAUDIODEST;


// IAudioDestNotifySink

#undef   INTERFACE
#define  INTERFACE   IAudioDestNotifySink

// {ACB08C00-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudioDestNotifySink,
0xacb08c00, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudioDestNotifySink, IUnknown) {
    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IAudioDestNotifySink members
    STDMETHOD (AudioStop)       (THIS_ WORD) PURE;
    STDMETHOD (AudioStart)      (THIS) PURE;
    STDMETHOD (FreeSpace)       (THIS_ DWORD, BOOL) PURE;
    STDMETHOD (BookMark)        (THIS_ DWORD, BOOL) PURE;
    };

typedef IAudioDestNotifySink FAR * PIAUDIODESTNOTIFYSINK;
```

```
// IAudioMultiMediaDevice

#undef   INTERFACE
#define  INTERFACE   IAudioMultiMediaDevice

// {B68AD320-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudioMultiMediaDevice,
0xb68ad320, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudioMultiMediaDevice, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioMultiMediaDevice members
   STDMETHOD (CustomMessage)  (THIS_ UINT, SDATA) PURE;
   STDMETHOD (DeviceNumGet)   (THIS_ DWORD*) PURE;
   STDMETHOD (DeviceNumSet)   (THIS_ DWORD) PURE;
   };

typedef IAudioMultiMediaDevice FAR * PIAUDIOMULTIMEDIADEVICE;



// IAudioSource
#undef   INTERFACE
#define  INTERFACE   IAudioSource

// {BC06A220-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudioSource,
0xbc06a220, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudioSource, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioSource members
   STDMETHOD (DataAvailable)  (THIS_ DWORD *, BOOL *) PURE;
   STDMETHOD (DataGet)        (THIS_ PVOID, DWORD, DWORD *) PURE;
   };

typedef IAudioSource FAR * PIAUDIOSOURCE;



// IAudioSourceInstrumented
#undef   INTERFACE
#define  INTERFACE   IAudioSourceInstrumented

// {D4023721-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(IID_IAudioSourceInstrumented,
0xd4023721, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (IAudioSourceInstrumented, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioSourceInstrumented members
   STDMETHOD (AudioSource)    (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (RegistryGet)    (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (RegistrySet)    (THIS_ PCWSTR) PURE;
   STDMETHOD (StateGet)       (THIS_ DWORD*) PURE;
   STDMETHOD (StateSet)       (THIS_ DWORD) PURE;
   STDMETHOD (WaveFileReadGet)(THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (WaveFileReadSet)(THIS_ PCWSTR) PURE;
```

```
   STDMETHOD (WaveFileWriteGet)(THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (WaveFileWriteSet)(THIS_ PCWSTR) PURE;
   };

typedef IAudioSourceInstrumented FAR * PIAUDIOSOURCEINSTRUMENTED;


// IAudioSourceSusp
#undef   INTERFACE
#define  INTERFACE   IAudioSourceSusp

// {D4123721-E4B9-11cf-8D56-00A0C9034A7E}
DEFINE_GUID(IID_IAudioSourceSusp,
0xd4123721, 0xe4b9, 0x11cf, 0x8d, 0x56, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (IAudioSourceSusp, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioSourceSusp members
   STDMETHOD (AudioSource)    (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Pause)          (THIS) PURE;
   STDMETHOD (Resume)         (THIS) PURE;
   };

typedef IAudioSourceSusp FAR * PIAUDIOSOURCESUSP;


// IAudioSourceNotifySink
#undef   INTERFACE
#define  INTERFACE   IAudioSourceNotifySink

// {C0BD9A80-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_IAudioSourceNotifySink,
0xc0bd9a80, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (IAudioSourceNotifySink, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioSourceNotifySink members
   STDMETHOD (AudioStop)      (THIS_ WORD) PURE;
   STDMETHOD (AudioStart)     (THIS) PURE;
   STDMETHOD (DataAvailable)  (THIS_ DWORD, BOOL) PURE;
   STDMETHOD (Overflow)       (THIS_ DWORD) PURE;
   };

typedef IAudioSourceNotifySink FAR * PIAUDIOSOURCENOTIFYSINK;

// IAudioFileNotifySink Interface
#undef   INTERFACE
#define  INTERFACE   IAudioFileNotifySink

// {492FE490-51E7-11b9-C000-FED6CBA3B1A9}
DEFINE_GUID( IID_IAudioFileNotifySink, 0x492fe490, 0x51e7, 0x11b9, 0xc0, 0x0, 0xfe, 0xd6, 0xcb, 0xa3,
0xb1, 0xa9);

DECLARE_INTERFACE_ (IAudioFileNotifySink, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioSourceNotifySink members
   STDMETHOD (FileBegin)      (THIS_ DWORD) PURE;
   STDMETHOD (FileEnd)        (THIS_ DWORD) PURE;
   STDMETHOD (QueueEmpty)     (THIS) PURE;
   STDMETHOD (Posn)           (THIS_ QWORD, QWORD) PURE;
```

```
   };

typedef IAudioFileNotifySink FAR * PIAUDIOFILENOTIFYSINK;


// IAudioFile Interface

#undef   INTERFACE
#define  INTERFACE   IAudioFile

// {FD7C2320-3D6D-11b9-C000-FED6CBA3B1A9}
DEFINE_GUID( IID_IAudioFile, 0xfd7c2320, 0x3d6d, 0x11b9, 0xc0, 0x0, 0xfe, 0xd6, 0xcb, 0xa3, 0xb1, 0xa9);

DECLARE_INTERFACE_ (IAudioFile, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioFile members
   STDMETHOD (Register) (THIS_ PIAUDIOFILENOTIFYSINK) PURE;
   STDMETHOD (Set) (THIS_ LPCWSTR pszFile, DWORD dwID) PURE;
   STDMETHOD (Add) (THIS_ LPCWSTR pszFile, DWORD dwID) PURE;
   STDMETHOD (Flush) (THIS) PURE;
   STDMETHOD (RealTimeSet) (THIS_ WORD wTime) PURE;
   STDMETHOD (RealTimeGet) (THIS_ WORD *pwTime) PURE;
   };

typedef IAudioFile FAR * PIAUDIOFILE;


// IAudioDirect Interface

#undef   INTERFACE
#define  INTERFACE   IAudioDirect

// {B9F11A94-90E3-11d0-8D77-00A0C9034A7E}
DEFINE_GUID(IID_IAudioDirect,
0xb9f11a94, 0x90e3, 0x11d0, 0x8d, 0x77, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (IAudioDirect, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IAudioDirect members
   STDMETHOD (Init) (THIS_ PVOID, REFIID) PURE;
   };

typedef IAudioDirect FAR * PIAUDIODIRECT;



/****************************************************************************
defines */
/* SRINFO */
#define  SRMI_NAMELEN                  SVFN_LEN

#define  SRSEQUENCE_DISCRETE           (0)
#define  SRSEQUENCE_CONTINUOUS          (1)
#define  SRSEQUENCE_WORDSPOT            (2)
#define  SRSEQUENCE_CONTCFGDISCDICT     (3)

#define  SRGRAM_CFG                    SETBIT(0)
#define  SRGRAM_DICTATION              SETBIT(1)
#define  SRGRAM_LIMITEDDOMAIN          SETBIT(2)

#define  SRFEATURE_INDEPSPEAKER        SETBIT(0)
#define  SRFEATURE_INDEPMICROPHONE     SETBIT(1)
#define  SRFEATURE_TRAINWORD           SETBIT(2)
```

```
#define  SRFEATURE_TRAINPHONETIC       SETBIT(3)
#define  SRFEATURE_WILDCARD            SETBIT(4)
#define  SRFEATURE_ANYWORD             SETBIT(5)
#define  SRFEATURE_PCOPTIMIZED         SETBIT(6)
#define  SRFEATURE_PHONEOPTIMIZED      SETBIT(7)
#define  SRFEATURE_GRAMLIST            SETBIT(8)
#define  SRFEATURE_GRAMLINK            SETBIT(9)
#define  SRFEATURE_MULTILINGUAL        SETBIT(10)
#define  SRFEATURE_GRAMRECURSIVE       SETBIT(11)
#define  SRFEATURE_IPAUNICODE          SETBIT(12)
#defineSRFEATURE_SINGLEINSTANCE      SETBIT(13)
#defineSRFEATURE_THREADSAFE          SETBIT(14)
#define  SRFEATURE_FIXEDAUDIO          SETBIT(15)
#define  SRFEATURE_IPAWORD             SETBIT(16)
#define  SRFEATURE_SAPI4               SETBIT(17)

#define  SRI_ILEXPRONOUNCE             SETBIT(0)
#define  SRI_ISRATTRIBUTES             SETBIT(1)
#define  SRI_ISRCENTRAL                SETBIT(2)
#define  SRI_ISRDIALOGS                SETBIT(3)
#define  SRI_ISRGRAMCOMMON             SETBIT(4)
#define  SRI_ISRGRAMCFG                SETBIT(5)
#define  SRI_ISRGRAMDICTATION          SETBIT(6)
#define  SRI_ISRGRAMINSERTIONGUI       SETBIT(7)
#define  SRI_ISRESBASIC                SETBIT(8)
#define  SRI_ISRESMERGE                SETBIT(9)
#define  SRI_ISRESAUDIO                SETBIT(10)
#define  SRI_ISRESCORRECTION           SETBIT(11)
#define  SRI_ISRESEVAL                 SETBIT(12)
#define  SRI_ISRESGRAPH                SETBIT(13)
#define  SRI_ISRESMEMORY               SETBIT(14)
#define  SRI_ISRESMODIFYGUI            SETBIT(15)
#define  SRI_ISRESSPEAKER              SETBIT(16)
#define  SRI_ISRSPEAKER                SETBIT(17)
#define  SRI_ISRESSCORES               SETBIT(18)
#define  SRI_ISRESAUDIOEX              SETBIT(19)
#define  SRI_ISRGRAMLEXPRON            SETBIT(20)
#define  SRI_ISRRESGRAPHEX             SETBIT(21)
#define  SRI_ILEXPRONOUNCE2            SETBIT(22)
#define  SRI_IATTRIBUTES               SETBIT(23)
#define  SRI_ISRSPEAKER2               SETBIT(24)
#define  SRI_ISRDIALOGS2               SETBIT(25)


// ISRGramCommon::TrainQuery
#define    SRGRAMQ_NONE                  0
#define    SRGRAMQ_GENERALTRAIN          1
#define    SRGRAMQ_PHRASE                2
#define    SRGRAMQ_DIALOG                3

// ISRGramNotifySink::PhraseFinish
#define    ISRNOTEFIN_RECOGNIZED         SETBIT(0)
#define    ISRNOTEFIN_THISGRAMMAR        SETBIT(1)
#define    ISRNOTEFIN_FROMTHISGRAMMAR    SETBIT(2)

// ISRGramNotifySink::Training
#define    SRGNSTRAIN_GENERAL            SETBIT(0)
#define    SRGNSTRAIN_GRAMMAR            SETBIT(1)
#define    SRGNSTRAIN_MICROPHONE         SETBIT(2)

// ISRNotifySink::AttribChange
#define    ISRNSAC_AUTOGAINENABLE        1
#define    ISRNSAC_THRESHOLD             2
#define    ISRNSAC_ECHO                  3
#define    ISRNSAC_ENERGYFLOOR           4
#define    ISRNSAC_MICROPHONE            5
#define    ISRNSAC_REALTIME              6
#define    ISRNSAC_SPEAKER               7
#define    ISRNSAC_TIMEOUT               8
#define    ISRNSAC_STARTLISTENING        9
#define    ISRNSAC_STOPLISTENING        10
```

```
/* Interference */
#define  SRMSGINT_NOISE                (0x0001)
#define  SRMSGINT_NOSIGNAL             (0x0002)
#define  SRMSGINT_TOOLOUD              (0x0003)
#define  SRMSGINT_TOOQUIET             (0x0004)
#define  SRMSGINT_AUDIODATA_STOPPED    (0x0005)
#define  SRMSGINT_AUDIODATA_STARTED    (0x0006)
#define  SRMSGINT_IAUDIO_STARTED       (0x0007)
#define  SRMSGINT_IAUDIO_STOPPED       (0x0008)

// Grammar header values
#define   SRHDRTYPE_CFG                 0
#define   SRHDRTYPE_LIMITEDDOMAIN       1
#define   SRHDRTYPE_DICTATION           2

#define   SRHDRFLAG_UNICODE             SETBIT(0)

// results objects defines
#define   SRRESCUE_COMMA                 1
#define   SRRESCUE_DECLARATIVEBEGIN      2
#define   SRRESCUE_DECLARATIVEEND        3
#define   SRRESCUE_IMPERATIVEBEGIN       4
#define   SRRESCUE_IMPERATIVEEND         5
#define   SRRESCUE_INTERROGATIVEBEGIN    6
#define   SRRESCUE_INTERROGATIVEEND      7
#define   SRRESCUE_NOISE                 8
#define   SRRESCUE_PAUSE                 9
#define   SRRESCUE_SENTENCEBEGIN         10
#define   SRRESCUE_SENTENCEEND           11
#define   SRRESCUE_UM                    12
#define   SRRESCUE_WILDCARD              13
#define   SRRESCUE_WORD                  14

/* SRCFGSYMBOL */
#define  SRCFG_STARTOPERATION          (1)
#define  SRCFG_ENDOPERATION            (2)
#define  SRCFG_WORD                    (3)
#define  SRCFG_RULE                    (4)
#define  SRCFG_WILDCARD                (5)
#define  SRCFG_LIST                    (6)

#define  SRCFGO_SEQUENCE               (1)
#define  SRCFGO_ALTERNATIVE            (2)
#define  SRCFGO_REPEAT                 (3)
#define  SRCFGO_OPTIONAL               (4)


// Grammar-chunk IDs
#define   SRCK_LANGUAGE                  1
#define   SRCKCFG_WORDS                  2
#define   SRCKCFG_RULES                  3
#define   SRCKCFG_EXPORTRULES            4
#define   SRCKCFG_IMPORTRULES            5
#define   SRCKCFG_LISTS                  6
#define   SRCKD_TOPIC                    7
#define   SRCKD_COMMON                   8
#define   SRCKD_GROUP                    9
#define   SRCKD_SAMPLE                   10
#define   SRCKLD_WORDS                   11
#define   SRCKLD_GROUP                   12
#define   SRCKLD_SAMPLE                  13
#define   SRCKD_WORDCOUNT                14
#define   SRCKD_NGRAM                    15

/* TrainQuery */
#define  SRTQEX_REQUIRED               (0x0000)
#define  SRTQEX_RECOMMENDED            (0x0001)

/* ISRResAudioEx */
#define  SRAUDIOTIMESTAMP_DEFAULT      ((QWORD) -1)

/* ISRResCorrection */
```

```
#define  SRCORCONFIDENCE_SOME          (0x0001)
#define  SRCORCONFIDENCE_VERY          (0x0002)

/* ISRResGraphEx */
DEFINE_GUID(SRGARC_ACOUSTICSCORE, 0xbf2f5b20, 0xb91d, 0x11d1, 0xb7, 0xbc, 0x0, 0x60, 0x97, 0x98, 0x41,
0xb4);
DEFINE_GUID(SRGARC_WORDTEXT, 0xbf2f5b21, 0xb91d, 0x11d1, 0xb7, 0xbc, 0x0, 0x60, 0x97, 0x98, 0x41, 0xb4);
DEFINE_GUID(SRGARC_NONLEXICALTEXT, 0x62ec00, 0xbaa0, 0x11d1, 0xb7, 0xbd, 0x0, 0x60, 0x97, 0x98, 0x41,
0xb4);
DEFINE_GUID(SRGARC_WORDPRONUNCIATION, 0xbf2f5b22, 0xb91d, 0x11d1, 0xb7, 0xbc, 0x0, 0x60, 0x97, 0x98,
0x41, 0xb4);
DEFINE_GUID(SRGNODE_TIME, 0xbf2f5b23, 0xb91d, 0x11d1, 0xb7, 0xbc, 0x0, 0x60, 0x97, 0x98, 0x41, 0xb4);

#define  SRGEX_ACOUSTICONLY            SETBIT(0)
#define  SRGEX_LMONLY                  SETBIT(1)
#define  SRGEX_ACOUSTICANDLM           SETBIT(2)

/* ISRResMemory constants */
#define  SRRESMEMKIND_AUDIO            SETBIT(0)
#define  SRRESMEMKIND_CORRECTION       SETBIT(1)
#define  SRRESMEMKIND_EVAL             SETBIT(2)
#define  SRRESMEMKIND_PHONEMEGRAPH     SETBIT(3)
#define  SRRESMEMKIND_WORDGRAPH        SETBIT(4)

// Attribute minimums and maximums
#define  SRATTR_MINAUTOGAIN              0
#define  SRATTR_MAXAUTOGAIN              100
#define  SRATTR_MINENERGYFLOOR           0
#define  SRATTR_MAXENERGYFLOOR           0xffff
#define  SRATTR_MINREALTIME              0
#define  SRATTR_MAXREALTIME              0xffffffff
#define  SRATTR_MINTHRESHOLD             0
#define  SRATTR_MAXTHRESHOLD             100
#define  SRATTR_MINTOINCOMPLETE          0
#define  SRATTR_MAXTOINCOMPLETE          0xffffffff
#define  SRATTR_MINTOCOMPLETE            0
#define  SRATTR_MAXTOCOMPLETE            0xffffffff


/****************************************************************************
typedefs */

typedef struct {
   DWORD    dwSize;
   DWORD    dwUniqueID;
   BYTE     abData[0];
   } SRCFGRULE, * PSRCFGRULE;



typedef struct {
   DWORD    dwSize;
   DWORD    dwRuleNum;
   WCHAR    szString[0];
   } SRCFGIMPRULEW, * PSRCFGIMPRULEW;

typedef struct {
   DWORD    dwSize;
   DWORD    dwRuleNum;
   CHAR     szString[0];
   } SRCFGIMPRULEA, * PSRCFGIMPRULEA;

#ifdef  _S_UNICODE
#define  SRCFGIMPRULE      SRCFGIMPRULEW
#define  PSRCFGIMPRULE     PSRCFGIMPRULEW
#else
#define  SRCFGIMPRULE      SRCFGIMPRULEA
#define  PSRCFGIMPRULE     PSRCFGIMPRULEA
#endif  // _S_UNICODE
```

```
typedef struct {
   DWORD    dwSize;
   DWORD    dwRuleNum;
   WCHAR    szString[0];
   } SRCFGXRULEW, * PSRCFGXRULEW;

typedef struct {
   DWORD    dwSize;
   DWORD    dwRuleNum;
   CHAR     szString[0];
   } SRCFGXRULEA, * PSRCFGXRULEA;

#ifdef  _S_UNICODE
#define  SRCFGXRULE     SRCFGXRULEW
#define  PSRCFGXRULE    PSRCFGXRULEW
#else
#define  SRCFGXRULE     SRCFGXRULEA
#define  PSRCFGXRULE    PSRCFGXRULEA
#endif  // _S_UNICODE


typedef struct {
   DWORD    dwSize;
   DWORD    dwListNum;
   WCHAR    szString[0];
   } SRCFGLISTW, * PSRCFGLISTW;

typedef struct {
   DWORD    dwSize;
   DWORD    dwListNum;
   CHAR     szString[0];
   } SRCFGLISTA, * PSRCFGLISTA;

#ifdef  _S_UNICODE
#define  SRCFGLIST      SRCFGLISTW
#define  PSRCFGLIST     PSRCFGLISTW
#else
#define  SRCFGLIST      SRCFGLISTA
#define  PSRCFGLIST     PSRCFGLISTA
#endif  // _S_UNICODE


typedef struct {
   WORD     wType;
   WORD     wProbability;
   DWORD    dwValue;
   } SRCFGSYMBOL, * PSRCFGSYMBOL;


typedef struct {
   DWORD    dwSize;
   DWORD    dwWordNum;
   WCHAR    szWord[0];
   } SRWORDW, * PSRWORDW;

typedef struct {
   DWORD    dwSize;
   DWORD    dwWordNum;
   CHAR     szWord[0];
   } SRWORDA, * PSRWORDA;

#ifdef  _S_UNICODE
#define  SRWORD      SRWORDW
#define  PSRWORD     PSRWORDW
#else
#define  SRWORD      SRWORDA
#define  PSRWORD     PSRWORDA
#endif  // _S_UNICODE
```

```
typedef struct {
   DWORD    dwSize;
   BYTE     abWords[0];
   } SRPHRASEW, * PSRPHRASEW;

typedef struct {
   DWORD    dwSize;
   BYTE     abWords[0];
   } SRPHRASEA, * PSRPHRASEA;

#ifdef  _S_UNICODE
#define  SRPHRASE    SRPHRASEW
#define  PSRPHRASE   PSRPHRASEW
#else
#define  SRPHRASE    SRPHRASEA
#define  PSRPHRASE   PSRPHRASEA
#endif  // _S_UNICODE


typedef struct {
   DWORD      dwType;
   DWORD      dwFlags;
   } SRHEADER, *PSRHEADER;

typedef struct {
   DWORD      dwChunkID;
   DWORD      dwChunkSize;
   BYTE       avInfo[0];
   } SRCHUNK, *PSRCHUNK;


typedef struct {
   GUID       gEngineID;
   WCHAR      szMfgName[SRMI_NAMELEN];
   WCHAR      szProductName[SRMI_NAMELEN];
   GUID       gModeID;
   WCHAR      szModeName[SRMI_NAMELEN];
   LANGUAGEW  language;
   DWORD      dwSequencing;
   DWORD      dwMaxWordsVocab;
   DWORD      dwMaxWordsState;
   DWORD      dwGrammars;
   DWORD      dwFeatures;
   DWORD      dwInterfaces;
   DWORD      dwEngineFeatures;
   } SRMODEINFOW, * PSRMODEINFOW;

typedef struct {
   GUID       gEngineID;
   CHAR       szMfgName[SRMI_NAMELEN];
   CHAR       szProductName[SRMI_NAMELEN];
   GUID       gModeID;
   CHAR       szModeName[SRMI_NAMELEN];
   LANGUAGEA  language;
   DWORD      dwSequencing;
   DWORD      dwMaxWordsVocab;
   DWORD      dwMaxWordsState;
   DWORD      dwGrammars;
   DWORD      dwFeatures;
   DWORD      dwInterfaces;
   DWORD      dwEngineFeatures;
   } SRMODEINFOA, * PSRMODEINFOA;

#ifdef  _S_UNICODE
#define  SRMODEINFO     SRMODEINFOW
#define  PSRMODEINFO    PSRMODEINFOW
#else
#define  SRMODEINFO     SRMODEINFOA
```

```
#define  PSRMODEINFO    PSRMODEINFOA
#endif  // _S_UNICODE


typedef struct {
   DWORD      dwEngineID;
   DWORD      dwMfgName;
   DWORD      dwProductName;
   DWORD      dwModeID;
   DWORD      dwModeName;
   DWORD      dwLanguage;
   DWORD      dwDialect;
   DWORD      dwSequencing;
   DWORD      dwMaxWordsVocab;
   DWORD      dwMaxWordsState;
   DWORD      dwGrammars;
   DWORD      dwFeatures;
   DWORD      dwInterfaces;
   DWORD      dwEngineFeatures;
   } SRMODEINFORANK, * PSRMODEINFORANK;



// speech recognition enumeration sharing object
typedef struct {
   QWORD        qwInstanceID;
   DWORD        dwDeviceID;
   SRMODEINFOW  srModeInfo;
} SRSHAREW, * PSRSHAREW;

typedef struct {
   QWORD        qwInstanceID;
   DWORD        dwDeviceID;
   SRMODEINFOA  srModeInfo;
} SRSHAREA, * PSRSHAREA;

#ifdef  _S_UNICODE
#define  SRSHARE    SRSHAREW
#define  PSRSHARE   PSRSHAREW
#else
#define  SRSHARE    SRSHAREA
#define  PSRSHARE   PSRSHAREA
#endif  // _S_UNICODE



// ISRCentral::GrammarLoad
typedef enum {
   SRGRMFMT_CFG = 0x0000,
   SRGRMFMT_LIMITEDDOMAIN = 0x0001,
   SRGRMFMT_DICTATION = 0x0002,
   SRGRMFMT_CFGNATIVE = 0x8000,
   SRGRMFMT_LIMITEDDOMAINNATIVE = 0x8001,
   SRGRMFMT_DICTATIONNATIVE = 0x8002,
   SRGRMFMT_DRAGONNATIVE1 = 0x8101,
   SRGRMFMT_DRAGONNATIVE2 = 0x8102,
   SRGRMFMT_DRAGONNATIVE3 = 0x8103
   } SRGRMFMT, * PSRGRMFMT;

// SRCKD_NGRAM
typedef struct {
    QWORD   dwTotalCounts;
    DWORD   dwNumWordsClasses;
    DWORD   dwWordClassNameOffset;
    DWORD   dwNumContextGroups;
    DWORD   dwContextGroupOffset;
    DWORD   dwNumClasses;
    DWORD   dwClassOffset;
    BYTE    bBitsPerWord;
    BYTE    abFiller[3];
```

```
    DWORD   adwProbability[256];
    } NGRAMHDR, * PNGRAMHDR;

typedef struct {
    DWORD   dwSize;
    QWORD   dwTotalCounts;
    DWORD   adwWordClassID[4];
    DWORD   dwNumSequential;
    DWORD   dwNumRandomAccess;
    BYTE    bBackOffProbToken;
    BYTE    abFiller[3];
    // WORD or DWORD awRandomAccessWordsClasses[dwNumRandomAccess];
    // BYTE abProbToken[dwNumSequential+dwNumRandomAccess];
    } NGRAMCGHDR, *PNGRAMCGHDR;


/****************************************************************************
Class IDs */


// {E02D16C0-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(CLSID_SREnumerator,
0xe02d16c0, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);


/****************************************************************************
interfaces */


/*
 * ISRAttributes
 */

#undef   INTERFACE
#define  INTERFACE   ISRAttributesW

DEFINE_GUID(IID_ISRAttributesW, 0x68A33AA0L, 0x44CD, 0x101B, 0x90, 0xA8, 0x00, 0xAA, 0x00, 0x3E, 0x4B,
0x50);

DECLARE_INTERFACE_ (ISRAttributesW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRAttributesW members
   STDMETHOD (AutoGainEnableGet) (THIS_ DWORD *) PURE;
   STDMETHOD (AutoGainEnableSet) (THIS_ DWORD) PURE;
   STDMETHOD (EchoGet)           (THIS_ BOOL *) PURE;
   STDMETHOD (EchoSet)           (THIS_ BOOL) PURE;
   STDMETHOD (EnergyFloorGet)    (THIS_ WORD *) PURE;
   STDMETHOD (EnergyFloorSet)    (THIS_ WORD) PURE;
   STDMETHOD (MicrophoneGet)     (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (MicrophoneSet)     (THIS_ PCWSTR) PURE;
   STDMETHOD (RealTimeGet)       (THIS_ DWORD *) PURE;
   STDMETHOD (RealTimeSet)       (THIS_ DWORD) PURE;
   STDMETHOD (SpeakerGet)        (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (SpeakerSet)        (THIS_ PCWSTR) PURE;
   STDMETHOD (TimeOutGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TimeOutSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ThresholdGet)      (THIS_ DWORD *) PURE;
   STDMETHOD (ThresholdSet)      (THIS_ DWORD) PURE;
   };

typedef ISRAttributesW FAR * PISRATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   ISRAttributesA
```

```
DEFINE_GUID(IID_ISRAttributesA, 0x2F26B9C1L, 0xDB31, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRAttributesA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRAttributesA members
   STDMETHOD (AutoGainEnableGet) (THIS_ DWORD *) PURE;
   STDMETHOD (AutoGainEnableSet) (THIS_ DWORD) PURE;
   STDMETHOD (EchoGet)           (THIS_ BOOL *) PURE;
   STDMETHOD (EchoSet)           (THIS_ BOOL) PURE;
   STDMETHOD (EnergyFloorGet)    (THIS_ WORD *) PURE;
   STDMETHOD (EnergyFloorSet)    (THIS_ WORD) PURE;
   STDMETHOD (MicrophoneGet)     (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (MicrophoneSet)     (THIS_ PCSTR) PURE;
   STDMETHOD (RealTimeGet)       (THIS_ DWORD *) PURE;
   STDMETHOD (RealTimeSet)       (THIS_ DWORD) PURE;
   STDMETHOD (SpeakerGet)        (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (SpeakerSet)        (THIS_ PCSTR) PURE;
   STDMETHOD (TimeOutGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TimeOutSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ThresholdGet)      (THIS_ DWORD *) PURE;
   STDMETHOD (ThresholdSet)      (THIS_ DWORD) PURE;
   };

typedef ISRAttributesA FAR * PISRATTRIBUTESA;


#ifdef _S_UNICODE
 #define ISRAttributes        ISRAttributesW
 #define IID_ISRAttributes    IID_ISRAttributesW
 #define PISRATTRIBUTES       PISRATTRIBUTESW

#else
 #define ISRAttributes        ISRAttributesA
 #define IID_ISRAttributes    IID_ISRAttributesA
 #define PISRATTRIBUTES       PISRATTRIBUTESA

#endif // _S_UNICODE

/*
 * ISRCentral
 */

#undef   INTERFACE
#define  INTERFACE   ISRCentralW

DEFINE_GUID(IID_ISRCentralW, 0xB9BD3860L, 0x44DB, 0x101B, 0x90, 0xA8, 0x00, 0xAA, 0x00, 0x3E, 0x4B,
0x50);

DECLARE_INTERFACE_ (ISRCentralW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRCentralW members
   STDMETHOD (ModeGet)          (THIS_ PSRMODEINFOW) PURE;
   STDMETHOD (GrammarLoad)      (THIS_ SRGRMFMT, SDATA, PVOID, IID, LPUNKNOWN *) PURE;
   STDMETHOD (Pause)            (THIS) PURE;
   STDMETHOD (PosnGet)          (THIS_ PQWORD) PURE;
   STDMETHOD (Resume)           (THIS) PURE;
   STDMETHOD (ToFileTime)       (THIS_ PQWORD, FILETIME *) PURE;
   STDMETHOD (Register)         (THIS_ PVOID, IID, DWORD*) PURE;
   STDMETHOD (UnRegister)       (THIS_ DWORD) PURE;
   };
```

```
typedef ISRCentralW FAR * PISRCENTRALW;


#undef   INTERFACE
#define  INTERFACE   ISRCentralA

DEFINE_GUID(IID_ISRCentralA, 0x2F26B9C2L, 0xDB31, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRCentralA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRCentralA members
   STDMETHOD (ModeGet)        (THIS_ PSRMODEINFOA) PURE;
   STDMETHOD (GrammarLoad)    (THIS_ SRGRMFMT, SDATA, PVOID, IID, LPUNKNOWN *) PURE;
   STDMETHOD (Pause)          (THIS) PURE;
   STDMETHOD (PosnGet)        (THIS_ PQWORD) PURE;
   STDMETHOD (Resume)         (THIS) PURE;
   STDMETHOD (ToFileTime)     (THIS_ PQWORD, FILETIME *) PURE;
   STDMETHOD (Register)       (THIS_ PVOID, IID, DWORD*) PURE;
   STDMETHOD (UnRegister)     (THIS_ DWORD) PURE;
   };

typedef ISRCentralA FAR * PISRCENTRALA;


#ifdef _S_UNICODE
 #define ISRCentral           ISRCentralW
 #define IID_ISRCentral       IID_ISRCentralW
 #define PISRCENTRAL          PISRCENTRALW

#else
 #define ISRCentral           ISRCentralA
 #define IID_ISRCentral       IID_ISRCentralA
 #define PISRCENTRAL          PISRCENTRALA

#endif   // _S_UNICODE



/*
 * ISRDialogs
 */

#undef   INTERFACE
#define  INTERFACE   ISRDialogsW

DEFINE_GUID(IID_ISRDialogsW, 0xBCFB4C60L, 0x44DB, 0x101B, 0x90, 0xA8, 0x00, 0xAA, 0x00, 0x3E, 0x4B,
0x50);

DECLARE_INTERFACE_ (ISRDialogsW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRDialogsW members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainMicDlg)    (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainGeneralDlg)(THIS_ HWND, PCWSTR) PURE;
   };

typedef ISRDialogsW FAR * PISRDIALOGSW;
```

```
#undef   INTERFACE
#define  INTERFACE   ISRDialogsA

DEFINE_GUID(IID_ISRDialogsA, 0x05EB6C60L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRDialogsA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRDialogsA members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainMicDlg)    (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainGeneralDlg)(THIS_ HWND, PCSTR) PURE;
   };

typedef ISRDialogsA FAR * PISRDIALOGSA;


#ifdef _S_UNICODE
 #define ISRDialogs        ISRDialogsW
 #define IID_ISRDialogs    IID_ISRDialogsW
 #define PISRDIALOGS       PISRDIALOGSW

#else
 #define ISRDialogs        ISRDialogsA
 #define IID_ISRDialogs    IID_ISRDialogsA
 #define PISRDIALOGS       PISRDIALOGSA

#endif


/*
 * ISRDialogs2
 */

#undef   INTERFACE
#define  INTERFACE   ISRDialogs2W

// {9B445330-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISRDialogs2W,
0x9b445330, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISRDialogs2W, ISRDialogsW) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRDialogs2W members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainMicDlg)    (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainGeneralDlg)(THIS_ HWND, PCWSTR) PURE;

   STDMETHOD (TrainPhrasesDlg)(THIS_ HWND, PCWSTR, PCWSTR) PURE;
   };

typedef ISRDialogs2W FAR * PISRDIALOGS2W;


#undef   INTERFACE
#define  INTERFACE   ISRDialogs2A

// {9B445331-E39F-11d1-BED7-006008317CE8}
```

```
DEFINE_GUID(IID_ISRDialogs2A,
0x9b445331, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISRDialogs2A, ISRDialogsA) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRDialogs2A members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainMicDlg)    (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainGeneralDlg)(THIS_ HWND, PCSTR) PURE;

   STDMETHOD (TrainPhrasesDlg)(THIS_ HWND, PCSTR, PCSTR) PURE;
   };

typedef ISRDialogs2A FAR * PISRDIALOGS2A;


#ifdef _S_UNICODE
 #define ISRDialogs2        ISRDialogs2W
 #define IID_ISRDialogs2    IID_ISRDialogs2W
 #define PISRDIALOGS2       PISRDIALOGS2W

#else
 #define ISRDialogs2        ISRDialogs2A
 #define IID_ISRDialogs2    IID_ISRDialogs2A
 #define PISRDIALOGS2       PISRDIALOGS2A

#endif


/*
 *  ISREnum
 */

#undef   INTERFACE
#define  INTERFACE   ISREnumW

DEFINE_GUID(IID_ISREnumW, 0xBFA9F1A0L, 0x44DB, 0x101B, 0x90, 0xA8, 0x00, 0xAA, 0x00, 0x3E, 0x4B, 0x50);

DECLARE_INTERFACE_ (ISREnumW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISREnumW members
   STDMETHOD (Next)           (THIS_ ULONG, PSRMODEINFOW, ULONG *) PURE;
   STDMETHOD (Skip)           (THIS_ ULONG) PURE;
   STDMETHOD (Reset)          (THIS) PURE;
   STDMETHOD (Clone)          (THIS_ ISREnumW * FAR *) PURE;
   STDMETHOD (Select)         (THIS_ GUID, PISRCENTRALW *, LPUNKNOWN) PURE;
   };

typedef ISREnumW FAR * PISRENUMW;


#undef   INTERFACE
#define  INTERFACE   ISREnumA

DEFINE_GUID(IID_ISREnumA, 0x05EB6C61L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISREnumA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
```

```
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISREnumA members
   STDMETHOD (Next)            (THIS_ ULONG, PSRMODEINFOA, ULONG *) PURE;
   STDMETHOD (Skip)            (THIS_ ULONG) PURE;
   STDMETHOD (Reset)           (THIS) PURE;
   STDMETHOD (Clone)           (THIS_ ISREnumA * FAR *) PURE;
   STDMETHOD (Select)          (THIS_ GUID, PISRCENTRALA *, LPUNKNOWN) PURE;
   };

typedef ISREnumA FAR * PISRENUMA;


#ifdef _S_UNICODE
 #define ISREnum           ISREnumW
 #define IID_ISREnum       IID_ISREnumW
 #define PISRENUM          PISRENUMW

#else
 #define ISREnum           ISREnumA
 #define IID_ISREnum       IID_ISREnumA
 #define PISRENUM          PISRENUMA

#endif



/*
 * ISRFind
 */

#undef   INTERFACE
#define  INTERFACE   ISRFindW

DEFINE_GUID(IID_ISRFindW, 0xC2835060L, 0x44DB, 0x101B, 0x90, 0xA8, 0x00, 0xAA, 0x00, 0x3E, 0x4B, 0x50);

DECLARE_INTERFACE_ (ISRFindW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRFindW members
   STDMETHOD (Find)            (THIS_ PSRMODEINFOW, PSRMODEINFORANK, PSRMODEINFOW) PURE;
   STDMETHOD (Select)          (THIS_ GUID, PISRCENTRALW *, LPUNKNOWN) PURE;
   };

typedef ISRFindW FAR * PISRFINDW;


#undef   INTERFACE
#define  INTERFACE   ISRFindA

DEFINE_GUID(IID_ISRFindA, 0x05EB6C62L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRFindA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRFindA members
   STDMETHOD (Find)            (THIS_ PSRMODEINFOA, PSRMODEINFORANK, PSRMODEINFOA) PURE;
   STDMETHOD (Select)          (THIS_ GUID, PISRCENTRALA *, LPUNKNOWN) PURE;
   };

typedef ISRFindA FAR * PISRFINDA;


#ifdef _S_UNICODE
```

```
 #define ISRFind            ISRFindW
 #define IID_ISRFind        IID_ISRFindW
 #define PISRFIND           PISRFINDW

#else
 #define ISRFind            ISRFindA
 #define IID_ISRFind        IID_ISRFindA
 #define PISRFIND           PISRFINDA

#endif


/*
 * ISRGramLexPron
 */

#undef   INTERFACE
#define  INTERFACE   ISRGramLexPronW

// {7EB4B631-19A7-11d1-AF95-0000F81E880D}
DEFINE_GUID(IID_ISRGramLexPronW, 0x7eb4b631, 0x19a7, 0x11d1, 0xaf, 0x95, 0x0, 0x0, 0xf8, 0x1e, 0x88,
0xd);

DECLARE_INTERFACE_ (ISRGramLexPronW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramLexPronW members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCWSTR, PCWSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD, BOOL) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCWSTR, WORD, PWSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCWSTR, WORD) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCWSTR, PCWSTR) PURE;
   };

typedef ISRGramLexPronW FAR * PISRGRAMLEXPRONW;


#undef   INTERFACE
#define  INTERFACE   ISRGramLexPronA

// {7EB4B632-19A7-11d1-AF95-0000F81E880D}
DEFINE_GUID(IID_ISRGramLexPronA, 0x7eb4b632, 0x19a7, 0x11d1, 0xaf, 0x95, 0x0, 0x0, 0xf8, 0x1e, 0x88,
0xd);

DECLARE_INTERFACE_ (ISRGramLexPronA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // LexPronA members
   STDMETHOD (Add)            (THIS_ VOICECHARSET, PCSTR, PCSTR,
                  VOICEPARTOFSPEECH, PVOID, DWORD, BOOL) PURE;
   STDMETHOD (Get)            (THIS_ VOICECHARSET, PCSTR, WORD, PSTR,
                  DWORD, DWORD *, VOICEPARTOFSPEECH *, PVOID,
                  DWORD, DWORD *) PURE;
   STDMETHOD (Remove)         (THIS_ PCSTR, WORD) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCSTR, PCSTR) PURE;
   };

typedef ISRGramLexPronA FAR * PISRGRAMLEXPRONA;


#ifdef _S_UNICODE
 #define ISRGramLexPron        ISRGramLexPronW
```

```
 #define IID_ISRGramLexPron     IID_ISRGramLexPronW
 #define PISRGRAMLEXPRON        PISRGRAMLEXPRONW

#else
 #define ISRGramLexPron         ISRGramLexPronA
 #define IID_ISRGramLexPron     IID_ISRGramLexPronA
 #define PISRGRAMLEXPRON        PISRGRAMLEXPRONA

#endif   // _S_UNICODE


/*
 * ISRGramCommon
 */

#undef   INTERFACE
#define  INTERFACE   ISRGramCommonW

DEFINE_GUID(IID_ISRGramCommonW, 0xe8c3e160, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b,
0x50);

DECLARE_INTERFACE_ (ISRGramCommonW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // ISRGramCommonW members
   STDMETHOD (Activate)          (THIS_ HWND, BOOL, PCWSTR) PURE;
   STDMETHOD (Archive)           (THIS_ BOOL, PVOID, DWORD, DWORD *) PURE;
   STDMETHOD (BookMark)          (THIS_ QWORD, DWORD) PURE;
   STDMETHOD (Deactivate)        (THIS_ PCWSTR) PURE;
   STDMETHOD (DeteriorationGet)  (THIS_ DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (DeteriorationSet)  (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TrainDlg)          (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainPhrase)       (THIS_ DWORD, PSDATA) PURE;
   STDMETHOD (TrainQuery)        (THIS_ DWORD *) PURE;
   };

typedef ISRGramCommonW FAR * PISRGRAMCOMMONW;


#undef   INTERFACE
#define  INTERFACE   ISRGramCommonA

DEFINE_GUID(IID_ISRGramCommonA, 0x05EB6C63L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRGramCommonA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // ISRGramCommonA members
   STDMETHOD (Activate)          (THIS_ HWND, BOOL, PCSTR) PURE;
   STDMETHOD (Archive)           (THIS_ BOOL, PVOID, DWORD, DWORD *) PURE;
   STDMETHOD (BookMark)          (THIS_ QWORD, DWORD) PURE;
   STDMETHOD (Deactivate)        (THIS_ PCSTR) PURE;
   STDMETHOD (DeteriorationGet)  (THIS_ DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (DeteriorationSet)  (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TrainDlg)          (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainPhrase)       (THIS_ DWORD, PSDATA) PURE;
   STDMETHOD (TrainQuery)        (THIS_ DWORD *) PURE;
   };

typedef ISRGramCommonA FAR * PISRGRAMCOMMONA;


#ifdef _S_UNICODE
```

```
 #define ISRGramCommon         ISRGramCommonW
 #define IID_ISRGramCommon     IID_ISRGramCommonW
 #define PISRGRAMCOMMON        PISRGRAMCOMMONW

#else
 #define ISRGramCommon         ISRGramCommonA
 #define IID_ISRGramCommon     IID_ISRGramCommonA
 #define PISRGRAMCOMMON        PISRGRAMCOMMONA

#endif



/*
 * ISRGramCFG
 */

#undef   INTERFACE
#define  INTERFACE   ISRGramCFGW

DEFINE_GUID(IID_ISRGramCFGW, 0xecc0b180, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (ISRGramCFGW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramCFGW members
   STDMETHOD (LinkQuery)      (THIS_ PCWSTR, BOOL *) PURE;
   STDMETHOD (ListAppend)     (THIS_ PCWSTR, SDATA) PURE;
   STDMETHOD (ListGet)        (THIS_ PCWSTR, PSDATA) PURE;
   STDMETHOD (ListRemove)     (THIS_ PCWSTR, SDATA) PURE;
   STDMETHOD (ListSet)        (THIS_ PCWSTR, SDATA) PURE;
   STDMETHOD (ListQuery)      (THIS_ PCWSTR, BOOL *) PURE;
   };

typedef ISRGramCFGW FAR * PISRGRAMCFGW;


#undef   INTERFACE
#define  INTERFACE   ISRGramCFGA

DEFINE_GUID(IID_ISRGramCFGA, 0x05EB6C64L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRGramCFGA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramCFGA members
   STDMETHOD (LinkQuery)      (THIS_ PCSTR, BOOL *) PURE;
   STDMETHOD (ListAppend)     (THIS_ PCSTR, SDATA) PURE;
   STDMETHOD (ListGet)        (THIS_ PCSTR, PSDATA) PURE;
   STDMETHOD (ListRemove)     (THIS_ PCSTR, SDATA) PURE;
   STDMETHOD (ListSet)        (THIS_ PCSTR, SDATA) PURE;
   STDMETHOD (ListQuery)      (THIS_ PCSTR, BOOL *) PURE;
   };

typedef ISRGramCFGA FAR * PISRGRAMCFGA;


#ifdef _S_UNICODE
 #define ISRGramCFG         ISRGramCFGW
 #define IID_ISRGramCFG     IID_ISRGramCFGW
 #define PISRGRAMCFG        PISRGRAMCFGW

#else
```

```
 #define ISRGramCFG         ISRGramCFGA
 #define IID_ISRGramCFG     IID_ISRGramCFGA
 #define PISRGRAMCFG        PISRGRAMCFGA

#endif


/*
 * ISRGramDictation
 */

#undef   INTERFACE
#define  INTERFACE   ISRGramDictationW

DEFINE_GUID(IID_ISRGramDictationW, 0x090CD9A3, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRGramDictationW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramDictationW members
   STDMETHOD (Context)        (THIS_ PCWSTR, PCWSTR) PURE;
   STDMETHOD (Hint)           (THIS_ PCWSTR) PURE;
   STDMETHOD (Words)          (THIS_ PCWSTR) PURE;
   };

typedef ISRGramDictationW FAR *PISRGRAMDICTATIONW;


#undef   INTERFACE
#define  INTERFACE   ISRGramDictationA

DEFINE_GUID(IID_ISRGramDictationA, 0x05EB6C65L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47,
0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRGramDictationA, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramDictationA members
   STDMETHOD (Context)        (THIS_ PCSTR, PCSTR) PURE;
   STDMETHOD (Hint)           (THIS_ PCSTR) PURE;
   STDMETHOD (Words)          (THIS_ PCSTR) PURE;
   };

typedef ISRGramDictationA FAR *PISRGRAMDICTATIONA;


#ifdef _S_UNICODE
 #define ISRGramDictation       ISRGramDictationW
 #define IID_ISRGramDictation   IID_ISRGramDictationW
 #define PISRGRAMDICTATION      PISRGRAMDICTATIONW

#else
 #define ISRGramDictation       ISRGramDictationA
 #define IID_ISRGramDictation   IID_ISRGramDictationA
 #define PISRGRAMDICTATION      PISRGRAMDICTATIONA

#endif


// ISRGramInsertionGUI
// This does not need an ANSI/UNICODE interface because no characters are passed
```

```
#undef   INTERFACE
#define  INTERFACE   ISRGramInsertionGUI

// {090CD9A4-DA1A-11CD-B3CA-00AA0047BA4F}
DEFINE_GUID(IID_ISRGramInsertionGUI,
0x090CD9A4, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRGramInsertionGUI, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramInsertionGUI members
   STDMETHOD (Hide)           (THIS) PURE;
   STDMETHOD (Move)           (THIS_ RECT) PURE;
   STDMETHOD (Show)           (THIS_ HWND) PURE;
   };

typedef ISRGramInsertionGUI FAR *PISRGRAMINSERTIONGUI;



/*
 * ISRGramNotifySink
 */

#undef   INTERFACE
#define  INTERFACE   ISRGramNotifySinkW

DEFINE_GUID(IID_ISRGramNotifySinkW,  0xf106bfa0, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b,
0x50);

DECLARE_INTERFACE_ (ISRGramNotifySinkW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramNotifySinkW members
   STDMETHOD (BookMark)       (THIS_ DWORD) PURE;
   STDMETHOD (Paused)         (THIS) PURE;
   STDMETHOD (PhraseFinish)   (THIS_ DWORD, QWORD, QWORD, PSRPHRASEW, LPUNKNOWN) PURE;
   STDMETHOD (PhraseHypothesis)(THIS_ DWORD, QWORD, QWORD, PSRPHRASEW, LPUNKNOWN) PURE;
   STDMETHOD (PhraseStart)    (THIS_ QWORD) PURE;
   STDMETHOD (ReEvaluate)     (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Training)       (THIS_ DWORD) PURE;
   STDMETHOD (UnArchive)      (THIS_ LPUNKNOWN) PURE;
   };

typedef ISRGramNotifySinkW FAR * PISRGRAMNOTIFYSINKW;


// ISRGramNotifySinkA
#undef   INTERFACE
#define  INTERFACE   ISRGramNotifySinkA

// {EFEEA350-CE5E-11cd-9D96-00AA002FC7C9}
DEFINE_GUID(IID_ISRGramNotifySinkA,
0xefeea350, 0xce5e, 0x11cd, 0x9d, 0x96, 0x0, 0xaa, 0x0, 0x2f, 0xc7, 0xc9);

DECLARE_INTERFACE_ (ISRGramNotifySinkA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRGramNotifySinkA members
   STDMETHOD (BookMark)       (THIS_ DWORD) PURE;
   STDMETHOD (Paused)         (THIS) PURE;
   STDMETHOD (PhraseFinish)   (THIS_ DWORD, QWORD, QWORD, PSRPHRASEA, LPUNKNOWN) PURE;
   STDMETHOD (PhraseHypothesis)(THIS_ DWORD, QWORD, QWORD, PSRPHRASEA, LPUNKNOWN) PURE;
```

```
   STDMETHOD (PhraseStart)     (THIS_ QWORD) PURE;
   STDMETHOD (ReEvaluate)      (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Training)        (THIS_ DWORD) PURE;
   STDMETHOD (UnArchive)       (THIS_ LPUNKNOWN) PURE;
   };

typedef ISRGramNotifySinkA FAR * PISRGRAMNOTIFYSINKA;


#ifdef _S_UNICODE
 #define ISRGramNotifySink       ISRGramNotifySinkW
 #define IID_ISRGramNotifySink   IID_ISRGramNotifySinkW
 #define PISRGRAMNOTIFYSINK      PISRGRAMNOTIFYSINKW

#else
 #define ISRGramNotifySink       ISRGramNotifySinkA
 #define IID_ISRGramNotifySink   IID_ISRGramNotifySinkA
 #define PISRGRAMNOTIFYSINK      PISRGRAMNOTIFYSINKA

#endif   // _S_UNICODE



// ISRNotifySink
// This does not need an ANSI/UNICODE interface because no characters are passed
#undef   INTERFACE
#define  INTERFACE   ISRNotifySink

DEFINE_GUID(IID_ISRNotifySink,
0x090CD9B0L, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRNotifySink, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRNotifySink members
   STDMETHOD (AttribChanged)  (THIS_ DWORD) PURE;
   STDMETHOD (Interference)   (THIS_ QWORD, QWORD, DWORD) PURE;
   STDMETHOD (Sound)          (THIS_ QWORD, QWORD) PURE;
   STDMETHOD (UtteranceBegin) (THIS_ QWORD) PURE;
   STDMETHOD (UtteranceEnd)   (THIS_ QWORD, QWORD) PURE;
   STDMETHOD (VUMeter)        (THIS_ QWORD, WORD) PURE;
   };

typedef ISRNotifySink FAR *PISRNOTIFYSINK;

// Just in case anyone uses the wide/ansi versions
#define ISRNotifySinkW       ISRNotifySink
#define IID_ISRNotifySinkW   IID_ISRNotifySink
#define PISRNOTIFYSINKW      PISRNOTIFYSINK
#define ISRNotifySinkA       ISRNotifySink
#define IID_ISRNotifySinkA   IID_ISRNotifySink
#define PISRNOTIFYSINKA      PISRNOTIFYSINK


// ISRNotifySink2
// This does not need an ANSI/UNICODE interface because no characters are passed
#undef   INTERFACE
#define  INTERFACE   ISRNotifySink2

// {9B445332-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISRNotifySink2,
0x9b445332, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISRNotifySink2, ISRNotifySink) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;
```

```
    // ISRNotifySink2 members
    STDMETHOD (AttribChanged)   (THIS_ DWORD) PURE;
    STDMETHOD (Interference)    (THIS_ QWORD, QWORD, DWORD) PURE;
    STDMETHOD (Sound)           (THIS_ QWORD, QWORD) PURE;
    STDMETHOD (UtteranceBegin)  (THIS_ QWORD) PURE;
    STDMETHOD (UtteranceEnd)    (THIS_ QWORD, QWORD) PURE;
    STDMETHOD (VUMeter)         (THIS_ QWORD, WORD) PURE;

    STDMETHOD (Training)        (THIS) PURE;
    STDMETHOD (Error)           (THIS_ LPUNKNOWN) PURE;
    STDMETHOD (Warning)         (THIS_ LPUNKNOWN) PURE;
    };

typedef ISRNotifySink2 FAR *PISRNOTIFYSINK2;


/*
 * ISRResBasic
 */

#undef   INTERFACE
#define  INTERFACE   ISRResBasicW

DEFINE_GUID(IID_ISRResBasicW, 0x090CD9A5, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResBasicW, IUnknown) {

    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ISRResBasicW members
    STDMETHOD (PhraseGet)       (THIS_ DWORD, PSRPHRASEW, DWORD,  DWORD *) PURE;
    STDMETHOD (Identify)        (THIS_ GUID *) PURE;
    STDMETHOD (TimeGet)         (THIS_ PQWORD, PQWORD) PURE;
    STDMETHOD (FlagsGet)        (THIS_ DWORD, DWORD *) PURE;
    };

typedef ISRResBasicW FAR *PISRRESBASICW;


#undef   INTERFACE
#define  INTERFACE   ISRResBasicA

DEFINE_GUID(IID_ISRResBasicA, 0x05EB6C66L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRResBasicA, IUnknown) {

    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ISRResBasicA members
    STDMETHOD (PhraseGet)       (THIS_ DWORD, PSRPHRASEA, DWORD,  DWORD *) PURE;
    STDMETHOD (Identify)        (THIS_ GUID *) PURE;
    STDMETHOD (TimeGet)         (THIS_ PQWORD, PQWORD) PURE;
    STDMETHOD (FlagsGet)        (THIS_ DWORD, DWORD *) PURE;
    };

typedef ISRResBasicA FAR *PISRRESBASICA;


#ifdef _S_UNICODE
 #define ISRResBasic            ISRResBasicW
 #define IID_ISRResBasic        IID_ISRResBasicW
 #define PISRRESBASIC           PISRRESBASICW

#else
 #define ISRResBasic            ISRResBasicA
```

```
 #define IID_ISRResBasic         IID_ISRResBasicA
 #define PISRRESBASIC            PISRRESBASICA

#endif   // _S_UNICODE


/*
 * ISRResScore
 * This does not need an ANSI/UNICODE interface because no characters are passed
 */

#undef INTERFACE
#define INTERFACE       ISRResScores


// {0B37F1E0-B8DE-11cf-B22E-00AA00A215ED}
DEFINE_GUID(IID_ISRResScores, 0xb37f1e0, 0xb8de, 0x11cf, 0xb2, 0x2e, 0x0, 0xaa, 0x0, 0xa2, 0x15, 0xed);

DECLARE_INTERFACE_ (ISRResScores, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface)      (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ISRResScores members
    STDMETHOD (GetPhraseScore) (THIS_ DWORD, long FAR *) PURE;
    STDMETHOD (GetWordScores)  (THIS_ DWORD, long FAR *, DWORD, LPDWORD) PURE;
};

typedef ISRResScores FAR* PISRRESSCORES;

// In case someone uses the A/W versions...

#define ISRResScoresW           ISRResScores
#define IID_ISRResScoresW       IID_ISRResScores
#define PISRRESSCORESW          PISRRESSCORES
#define ISRResScoresA           ISRResScores
#define IID_ISRResScoresA       IID_ISRResScores
#define PISRRESSCORESA          PISRRESSCORES



/*
 * ISRResMerge
 * This does not need an ANSI/UNICODE interface because no characters are passed
 */

#undef   INTERFACE
#define  INTERFACE   ISRResMerge

DEFINE_GUID(IID_ISRResMerge, 0x090CD9A6, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResMerge, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISRResMerge members
   STDMETHOD (Merge)           (THIS_ LPUNKNOWN, PIUNKNOWN ) PURE;
   STDMETHOD (Split)           (THIS_ QWORD, PIUNKNOWN , PIUNKNOWN ) PURE;
   };

typedef ISRResMerge FAR *PISRRESMERGE;



/*
 * ISRResAudio
 * This does not need an ANSI/UNICODE interface because no characters are passed
```

```
 */

#undef   INTERFACE
#define  INTERFACE   ISRResAudio

DEFINE_GUID(IID_ISRResAudio, 0x090CD9A7, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResAudio, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRResAudio members
   STDMETHOD (GetWAV)         (THIS_ PSDATA) PURE;
   };

typedef ISRResAudio FAR *PISRRESAUDIO;



/*
 * ISRResAudioEx
 * This does not need an ANSI/UNICODE interface because no characters are passed
 */

#undef   INTERFACE
#define  INTERFACE   ISRResAudioEx

DEFINE_GUID(IID_ISRResAudioEx, 0xaad74c30, 0x4b02, 0x11d1, 0xb7, 0x92, 0x0, 0x60, 0x97, 0x98, 0x41,
0xb4);

DECLARE_INTERFACE_ (ISRResAudioEx, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRResAudioEx members
   STDMETHOD (GetWAV)         (THIS_ PSDATA, QWORD, QWORD) PURE;
   };

typedef ISRResAudioEx FAR *PISRRESAUDIOEX;



/*
 * ISRResCorrection
 */

#undef   INTERFACE
#define  INTERFACE   ISRResCorrectionW

DEFINE_GUID(IID_ISRResCorrectionW, 0x090CD9A8L, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47,
0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResCorrectionW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRResCorrectionW members
   STDMETHOD (Correction)         (THIS_ PSRPHRASEW, WORD) PURE;
   STDMETHOD (Validate)           (THIS_ WORD) PURE;
   };

typedef ISRResCorrectionW FAR *PISRRESCORRECTIONW;
```

```
#undef   INTERFACE
#define  INTERFACE   ISRResCorrectionA

DEFINE_GUID(IID_ISRResCorrectionA, 0x05EB6C67L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47,
0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResCorrectionA, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRResCorrectionA members
   STDMETHOD (Correction)          (THIS_ PSRPHRASEA, WORD) PURE;
   STDMETHOD (Validate)            (THIS_ WORD) PURE;
   };

typedef ISRResCorrectionA FAR *PISRRESCORRECTIONA;


#ifdef _S_UNICODE
 #define ISRResCorrection        ISRResCorrectionW
 #define IID_ISRResCorrection    IID_ISRResCorrectionW
 #define PISRRESCORRECTION       PISRRESCORRECTIONW

#else
 #define ISRResCorrection        ISRResCorrectionA
 #define IID_ISRResCorrection    IID_ISRResCorrectionA
 #define PISRRESCORRECTION       PISRRESCORRECTIONA

#endif   // _S_UNICODE



// ISRResEval
// This does not need an ANSI/UNICODE interface because no characters are passed
#undef   INTERFACE
#define  INTERFACE   ISRResEval

// {90CD9A9-DA1A-11CD-B3CA-00AA0047BA4F}
DEFINE_GUID(IID_ISRResEval,
0x090CD9A9, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResEval, IUnknown) {
   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   //  SRResEval members
   STDMETHOD (ReEvaluate)     (THIS_ BOOL *) PURE;
   };

typedef ISRResEval FAR *PISRRESEVAL;



/*
 * ISRResGraph
 */

#undef   INTERFACE
#define  INTERFACE ISRResGraphW

DEFINE_GUID(IID_ISRResGraphW, 0x090CD9AA, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResGraphW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
```

```
    STDMETHOD_(ULONG,AddRef)        (THIS) PURE;
    STDMETHOD_(ULONG,Release)       (THIS) PURE;

    // ISRResGraphW members
    STDMETHOD (BestPathPhoneme)     (THIS_ DWORD, DWORD *, DWORD, DWORD *) PURE;
    STDMETHOD (BestPathWord)        (THIS_ DWORD, DWORD *, DWORD, DWORD *) PURE;
    STDMETHOD (GetPhonemeNode)      (THIS_ DWORD, PSRRESPHONEMENODE, PWCHAR,
                PWCHAR) PURE;
    STDMETHOD (GetWordNode)         (THIS_ DWORD, PSRRESWORDNODE, PSRWORDW, DWORD,
                DWORD *) PURE;
    STDMETHOD (PathScorePhoneme)    (THIS_ DWORD *, DWORD, LONG *) PURE;
    STDMETHOD (PathScoreWord)       (THIS_ DWORD *, DWORD, LONG *) PURE;
    };

typedef ISRResGraphW FAR *PISRRESGRAPHW;


#undef   INTERFACE
#define  INTERFACE ISRResGraphA

DEFINE_GUID(IID_ISRResGraphA, 0x05EB6C68L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRResGraphA, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface)      (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)        (THIS) PURE;
    STDMETHOD_(ULONG,Release)       (THIS) PURE;

    // ISRResGraphA members
    STDMETHOD (BestPathPhoneme)     (THIS_ DWORD, DWORD *, DWORD, DWORD *) PURE;
    STDMETHOD (BestPathWord)        (THIS_ DWORD, DWORD *, DWORD, DWORD *) PURE;
    STDMETHOD (GetPhonemeNode)      (THIS_ DWORD, PSRRESPHONEMENODE, PWCHAR,
                PCHAR) PURE;
    STDMETHOD (GetWordNode)         (THIS_ DWORD, PSRRESWORDNODE, PSRWORDA, DWORD,
                DWORD *) PURE;
    STDMETHOD (PathScorePhoneme)    (THIS_ DWORD *, DWORD, LONG *) PURE;
    STDMETHOD (PathScoreWord)       (THIS_ DWORD *, DWORD, LONG *) PURE;
    };

typedef ISRResGraphA FAR *PISRRESGRAPHA;


#ifdef _S_UNICODE
 #define ISRResGraph            ISRResGraphW
 #define IID_ISRResGraph        IID_ISRResGraphW
 #define PISRRESGRAPH           PISRRESGRAPHW

#else
 #define ISRResGraph            ISRResGraphA
 #define IID_ISRResGraph        IID_ISRResGraphA
 #define PISRRESGRAPH           PISRRESGRAPHA

#endif   // _S_UNICODE



/*
 * ISRResGraphEx
 */

#undef   INTERFACE
#define  INTERFACE ISRResGraphEx

DEFINE_GUID(IID_ISRResGraphEx, 0xbe8f63a0, 0xb915, 0x11d1, 0xb7, 0xbc, 0x0, 0x60, 0x97, 0x98, 0x41,
0xb4);

DECLARE_INTERFACE_ (ISRResGraphEx, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface)      (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
```

```
    STDMETHOD_(ULONG,AddRef)       (THIS) PURE;
    STDMETHOD_(ULONG,Release)      (THIS) PURE;

    // ISRResGraphEx members
    STDMETHOD (NodeStartGet)       (THIS_ DWORD *) PURE;
    STDMETHOD (NodeEndGet)         (THIS_ DWORD *) PURE;
    STDMETHOD (ArcEnum)            (THIS_ DWORD, DWORD *, DWORD, DWORD *, BOOL) PURE;
    STDMETHOD (NodeGet)            (THIS_ DWORD, DWORD *, BOOL) PURE;
    STDMETHOD (DWORDGet)           (THIS_ DWORD, GUID, DWORD *) PURE;
    STDMETHOD (DataGet)            (THIS_ DWORD, GUID, SDATA *) PURE;
    STDMETHOD (ScoreGet)           (THIS_ DWORD, DWORD *, DWORD, DWORD, DWORD, DWORD *) PURE;
    STDMETHOD (BestPathEnum)       (THIS_ DWORD, DWORD *, DWORD, DWORD *, DWORD, BOOL, SDATA *) PURE;
    STDMETHOD (GetAllArcs)         (THIS_ DWORD *, DWORD, DWORD *, DWORD *) PURE;
    STDMETHOD (GetAllNodes)        (THIS_ DWORD *, DWORD, DWORD *, DWORD *) PURE;
};

typedef ISRResGraphEx FAR *PISRRESGRAPHEX;


// ISRResMemory
// This does not need an ANSI/UNICODE interface because no characters are passed
#undef   INTERFACE
#define  INTERFACE   ISRResMemory

DEFINE_GUID(IID_ISRResMemory, 0x090CD9AB, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRResMemory, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
    STDMETHOD_(ULONG,Release)  (THIS) PURE;

    // ISRResMemory members
    STDMETHOD (Free)           (THIS_ DWORD) PURE;
    STDMETHOD (Get)            (THIS_ DWORD *, DWORD *) PURE;
    STDMETHOD (LockGet)        (THIS_ BOOL *) PURE;
    STDMETHOD (LockSet)        (THIS_ BOOL) PURE;
    };

typedef ISRResMemory FAR *PISRRESMEMORY;


// ISRResModifyGUI
// This does not need an ANSI/UNICODE interface because no characters are passed
#undef   INTERFACE
#define  INTERFACE   ISRResModifyGUI

DEFINE_GUID(IID_ISRResModifyGUI, 0x090CD9AC, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRResModifyGUI, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)       (THIS) PURE;
    STDMETHOD_(ULONG,Release)      (THIS) PURE;

    // ISRResModifyGUI members
    STDMETHOD (Hide)               (THIS) PURE;
    STDMETHOD (Move)               (THIS_ RECT *) PURE;
    STDMETHOD (Show)               (THIS_ HWND) PURE;
    };

typedef ISRResModifyGUI FAR *PISRRESMODIFYGUI;


/*
 * ISRResSpeakerW
```

```
 */

#undef   INTERFACE
#define  INTERFACE   ISRResSpeakerW

DEFINE_GUID(IID_ISRResSpeakerW, 0x090CD9AD, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRResSpeakerW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface)    (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // ISRResSpeakerW members
   STDMETHOD (Correction)        (THIS_ PCWSTR, WORD) PURE;
   STDMETHOD (Validate)          (THIS_ WORD) PURE;
   STDMETHOD (Identify)          (THIS_ DWORD, PWSTR, DWORD, DWORD *,
                 LONG *) PURE;
   STDMETHOD (IdentifyForFree)   (THIS_ BOOL *) PURE;
   };

typedef ISRResSpeakerW FAR *PISRRESSPEAKERW;


#undef   INTERFACE
#define  INTERFACE   ISRResSpeakerA

DEFINE_GUID(IID_ISRResSpeakerA, 0x05EB6C69L, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ISRResSpeakerA, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface)    (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // ISRResSpeakerA members
   STDMETHOD (Correction)        (THIS_ PCSTR, WORD) PURE;
   STDMETHOD (Validate)          (THIS_ WORD) PURE;
   STDMETHOD (Identify)          (THIS_ DWORD, PSTR, DWORD, DWORD *,
                 LONG *) PURE;
   STDMETHOD (IdentifyForFree)   (THIS_ BOOL *) PURE;
   };

typedef ISRResSpeakerA FAR *PISRRESSPEAKERA;


#ifdef _S_UNICODE
 #define ISRResSpeaker           ISRResSpeakerW
 #define IID_ISRResSpeaker       IID_ISRResSpeakerW
 #define PISRRESSPEAKER          PISRRESSPEAKERW

#else
 #define ISRResSpeaker           ISRResSpeakerA
 #define IID_ISRResSpeaker       IID_ISRResSpeakerA
 #define PISRRESSPEAKER          PISRRESSPEAKERA

#endif   // _S_UNICODE


/*
 * ISRSpeaker
 */

#undef   INTERFACE
#define  INTERFACE   ISRSpeakerW

DEFINE_GUID(IID_ISRSpeakerW, 0x090CD9AE, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);
```

```
DECLARE_INTERFACE_ (ISRSpeakerW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRSpeakerW members
   STDMETHOD (Delete)         (THIS_ PCWSTR) PURE;
   STDMETHOD (Enum)           (THIS_ PWSTR *, DWORD *) PURE;
   STDMETHOD (Merge)          (THIS_ PCWSTR, PVOID, DWORD) PURE;
   STDMETHOD (New)            (THIS_ PCWSTR) PURE;
   STDMETHOD (Query)          (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (Read)           (THIS_ PCWSTR, PVOID *, DWORD *) PURE;
   STDMETHOD (Revert)         (THIS_ PCWSTR) PURE;
   STDMETHOD (Select)         (THIS_ PCWSTR, BOOL) PURE;
   STDMETHOD (Write)          (THIS_ PCWSTR, PVOID, DWORD) PURE;
   };

typedef ISRSpeakerW FAR *PISRSPEAKERW;


#undef   INTERFACE
#define  INTERFACE   ISRSpeakerA

DEFINE_GUID(IID_ISRSpeakerA, 0x090CD9AF, 0xDA1A, 0x11CD, 0xB3, 0xCA, 0x0, 0xAA, 0x0, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ISRSpeakerA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRSpeakerA members
   STDMETHOD (Delete)         (THIS_ PCSTR) PURE;
   STDMETHOD (Enum)           (THIS_ PSTR *, DWORD *) PURE;
   STDMETHOD (Merge)          (THIS_ PCSTR, PVOID, DWORD) PURE;
   STDMETHOD (New)            (THIS_ PCSTR) PURE;
   STDMETHOD (Query)          (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (Read)           (THIS_ PCSTR, PVOID *, DWORD *) PURE;
   STDMETHOD (Revert)         (THIS_ PCSTR) PURE;
   STDMETHOD (Select)         (THIS_ PCSTR, BOOL) PURE;
   STDMETHOD (Write)          (THIS_ PCSTR, PVOID, DWORD) PURE;
   };

typedef ISRSpeakerA FAR *PISRSPEAKERA;


#ifdef _S_UNICODE
 #define ISRSpeaker             ISRSpeakerW
 #define IID_ISRSpeaker         IID_ISRSpeakerW
 #define PISRSPEAKER            PISRSPEAKERW

#else
 #define ISRSpeaker             ISRSpeakerA
 #define IID_ISRSpeaker         IID_ISRSpeakerA
 #define PISRSPEAKER            PISRSPEAKERA

#endif   // _S_UNICODE


/*
 * ISRSpeaker2
 */

#undef   INTERFACE
#define  INTERFACE   ISRSpeaker2W

// {9B445333-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISRSpeaker2W,
```

```
0x9b445333, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISRSpeaker2W, ISRSpeakerW) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRSpeaker2W members
   STDMETHOD (Delete)         (THIS_ PCWSTR) PURE;
   STDMETHOD (Enum)           (THIS_ PWSTR *, DWORD *) PURE;
   STDMETHOD (Merge)          (THIS_ PCWSTR, PVOID, DWORD) PURE;
   STDMETHOD (New)            (THIS_ PCWSTR) PURE;
   STDMETHOD (Query)          (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD (Read)           (THIS_ PCWSTR, PVOID *, DWORD *) PURE;
   STDMETHOD (Revert)         (THIS_ PCWSTR) PURE;
   STDMETHOD (Select)         (THIS_ PCWSTR, BOOL) PURE;
   STDMETHOD (Write)          (THIS_ PCWSTR, PVOID, DWORD) PURE;

   STDMETHOD (Commit)         (THIS) PURE;
   STDMETHOD (Rename)         (THIS_ PCWSTR, PCWSTR) PURE;
   STDMETHOD (GetChangedInfo) (THIS_ BOOL *, FILETIME*) PURE;
   };

typedef ISRSpeaker2W FAR *PISRSPEAKER2W;


#undef   INTERFACE
#define  INTERFACE   ISRSpeaker2A

// {9B445334-E39F-11d1-BED7-006008317CE8}
DEFINE_GUID(IID_ISRSpeaker2A,
0x9b445334, 0xe39f, 0x11d1, 0xbe, 0xd7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ISRSpeaker2A, ISRSpeakerA) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ISRSpeaker2A members
   STDMETHOD (Delete)         (THIS_ PCSTR) PURE;
   STDMETHOD (Enum)           (THIS_ PSTR *, DWORD *) PURE;
   STDMETHOD (Merge)          (THIS_ PCSTR, PVOID, DWORD) PURE;
   STDMETHOD (New)            (THIS_ PCSTR) PURE;
   STDMETHOD (Query)          (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (Read)           (THIS_ PCSTR, PVOID *, DWORD *) PURE;
   STDMETHOD (Revert)         (THIS_ PCSTR) PURE;
   STDMETHOD (Select)         (THIS_ PCSTR, BOOL) PURE;
   STDMETHOD (Write)          (THIS_ PCSTR, PVOID, DWORD) PURE;

   STDMETHOD (Commit)         (THIS) PURE;
   STDMETHOD (Rename)         (THIS_ PCSTR, PCSTR) PURE;
   STDMETHOD (GetChangedInfo) (THIS_ BOOL *, FILETIME*) PURE;
   };

typedef ISRSpeaker2A FAR *PISRSPEAKER2A;


#ifdef _S_UNICODE
 #define ISRSpeaker2              ISRSpeaker2W
 #define IID_ISRSpeaker2          IID_ISRSpeaker2W
 #define PISRSPEAKER2             PISRSPEAKER2W

#else
 #define ISRSpeaker2              ISRSpeaker2A
 #define IID_ISRSpeaker2          IID_ISRSpeaker2A
 #define PISRSPEAKER2             PISRSPEAKER2A

#endif   // _S_UNICODE
```

```
/*****************************************************************************
Low-Level text-to-speech API
*/


/*****************************************************************************
defines */

#define  TTSI_NAMELEN                    SVFN_LEN
#define  TTSI_STYLELEN                   SVFN_LEN

#define  GENDER_NEUTRAL                  (0)
#define  GENDER_FEMALE                   (1)
#define  GENDER_MALE                     (2)

#define  TTSFEATURE_ANYWORD              SETBIT(0)
#define  TTSFEATURE_VOLUME               SETBIT(1)
#define  TTSFEATURE_SPEED                SETBIT(2)
#define  TTSFEATURE_PITCH                SETBIT(3)
#define  TTSFEATURE_TAGGED               SETBIT(4)
#define  TTSFEATURE_IPAUNICODE           SETBIT(5)
#define  TTSFEATURE_VISUAL               SETBIT(6)
#define  TTSFEATURE_WORDPOSITION         SETBIT(7)
#define  TTSFEATURE_PCOPTIMIZED          SETBIT(8)
#define  TTSFEATURE_PHONEOPTIMIZED       SETBIT(9)
#define  TTSFEATURE_FIXEDAUDIO           SETBIT(10)
#defineTTSFEATURE_SINGLEINSTANCE      SETBIT(11)
#defineTTSFEATURE_THREADSAFE          SETBIT(12)
#defineTTSFEATURE_IPATEXTDATA         SETBIT(13)
#defineTTSFEATURE_PREFERRED           SETBIT(14)
#defineTTSFEATURE_TRANSPLANTED        SETBIT(15)
#defineTTSFEATURE_SAPI4               SETBIT(16)

#define  TTSI_ILEXPRONOUNCE              SETBIT(0)
#define  TTSI_ITTSATTRIBUTES             SETBIT(1)
#define  TTSI_ITTSCENTRAL                SETBIT(2)
#define  TTSI_ITTSDIALOGS                SETBIT(3)
#define  TTSI_ATTRIBUTES                 SETBIT(4)
#define  TTSI_IATTRIBUTES                 SETBIT(4)
#defineTTSI_ILEXPRONOUNCE2             SETBIT(5)

#define  TTSDATAFLAG_TAGGED              SETBIT(0)

#define   TTSBNS_ABORTED                    SETBIT(0)

// ITTSNotifySink
#define   TTSNSAC_REALTIME                0
#define   TTSNSAC_PITCH                   1
#define   TTSNSAC_SPEED                   2
#define   TTSNSAC_VOLUME                  3


#define   TTSNSHINT_QUESTION              SETBIT(0)
#define   TTSNSHINT_STATEMENT             SETBIT(1)
#define   TTSNSHINT_COMMAND               SETBIT(2)
#define   TTSNSHINT_EXCLAMATION           SETBIT(3)
#define   TTSNSHINT_EMPHASIS              SETBIT(4)


// Ages
#define  TTSAGE_BABY                     1
#define  TTSAGE_TODDLER                  3
#define  TTSAGE_CHILD                    6
#define  TTSAGE_ADOLESCENT               14
#define  TTSAGE_ADULT                    30
#define  TTSAGE_ELDERLY                  70

// Attribute minimums and maximums
#define  TTSATTR_MINPITCH                0
#define  TTSATTR_MAXPITCH                0xffff
```

```
#define  TTSATTR_MINREALTIME          0
#define  TTSATTR_MAXREALTIME          0xffffffff
#define  TTSATTR_MINSPEED             0
#define  TTSATTR_MAXSPEED             0xffffffff
#define  TTSATTR_MINVOLUME            0
#define  TTSATTR_MAXVOLUME            0xffffffff


/*****************************************************************************
typedefs */

typedef struct {
   BYTE     bMouthHeight;
   BYTE     bMouthWidth;
   BYTE     bMouthUpturn;
   BYTE     bJawOpen;
   BYTE     bTeethUpperVisible;
   BYTE     bTeethLowerVisible;
   BYTE     bTonguePosn;
   BYTE     bLipTension;
   } TTSMOUTH, *PTTSMOUTH;



typedef struct {
   GUID        gEngineID;
   WCHAR       szMfgName[TTSI_NAMELEN];
   WCHAR       szProductName[TTSI_NAMELEN];
   GUID        gModeID;
   WCHAR       szModeName[TTSI_NAMELEN];
   LANGUAGEW   language;
   WCHAR       szSpeaker[TTSI_NAMELEN];
   WCHAR       szStyle[TTSI_STYLELEN];
   WORD        wGender;
   WORD        wAge;
   DWORD       dwFeatures;
   DWORD       dwInterfaces;
   DWORD       dwEngineFeatures;
   } TTSMODEINFOW, *PTTSMODEINFOW;

typedef struct {
   GUID        gEngineID;
   CHAR        szMfgName[TTSI_NAMELEN];
   CHAR        szProductName[TTSI_NAMELEN];
   GUID        gModeID;
   CHAR        szModeName[TTSI_NAMELEN];
   LANGUAGEA   language;
   CHAR        szSpeaker[TTSI_NAMELEN];
   CHAR        szStyle[TTSI_STYLELEN];
   WORD        wGender;
   WORD        wAge;
   DWORD       dwFeatures;
   DWORD       dwInterfaces;
   DWORD       dwEngineFeatures;
   } TTSMODEINFOA, *PTTSMODEINFOA;

#ifdef _S_UNICODE
 #define TTSMODEINFO          TTSMODEINFOW
 #define PTTSMODEINFO         PTTSMODEINFOW

#else
 #define TTSMODEINFO          TTSMODEINFOA
 #define PTTSMODEINFO         PTTSMODEINFOA

#endif   // _S_UNICODE


typedef struct {
   DWORD       dwEngineID;
   DWORD       dwMfgName;
   DWORD       dwProductName;
```

```
   DWORD      dwModeID;
   DWORD      dwModeName;
   DWORD      dwLanguage;
   DWORD      dwDialect;
   DWORD      dwSpeaker;
   DWORD      dwStyle;
   DWORD      dwGender;
   DWORD      dwAge;
   DWORD      dwFeatures;
   DWORD      dwInterfaces;
   DWORD      dwEngineFeatures;
   } TTSMODEINFORANK, * PTTSMODEINFORANK;

/*****************************************************************************
Class IDs */
// {D67C0280-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(CLSID_TTSEnumerator,
0xd67c0280, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

/*****************************************************************************
interfaces */

// ITTSAttributes

#undef   INTERFACE
#define  INTERFACE   ITTSAttributesW

DEFINE_GUID(IID_ITTSAttributesW, 0x1287A280L, 0x4A47, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSAttributesW, IUnknown) {

// IUnknown members

   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

// ITTSAttributes members

   STDMETHOD (PitchGet)        (THIS_ WORD *) PURE;
   STDMETHOD (PitchSet)        (THIS_ WORD) PURE;
   STDMETHOD (RealTimeGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (RealTimeSet)     (THIS_ DWORD) PURE;
   STDMETHOD (SpeedGet)        (THIS_ DWORD *) PURE;
   STDMETHOD (SpeedSet)        (THIS_ DWORD) PURE;
   STDMETHOD (VolumeGet)       (THIS_ DWORD *) PURE;
   STDMETHOD (VolumeSet)       (THIS_ DWORD) PURE;
   };

typedef ITTSAttributesW FAR * PITTSATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   ITTSAttributesA

DEFINE_GUID(IID_ITTSAttributesA,
0x0FD6E2A1L, 0xE77D, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ITTSAttributesA, IUnknown) {

// IUnknown members

   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

// ITTSAttributes members

   STDMETHOD (PitchGet)        (THIS_ WORD *) PURE;
   STDMETHOD (PitchSet)        (THIS_ WORD) PURE;
   STDMETHOD (RealTimeGet)     (THIS_ DWORD *) PURE;
```

```
   STDMETHOD (RealTimeSet)     (THIS_ DWORD) PURE;
   STDMETHOD (SpeedGet)        (THIS_ DWORD *) PURE;
   STDMETHOD (SpeedSet)        (THIS_ DWORD) PURE;
   STDMETHOD (VolumeGet)       (THIS_ DWORD *) PURE;
   STDMETHOD (VolumeSet)       (THIS_ DWORD) PURE;
   };

typedef ITTSAttributesA FAR * PITTSATTRIBUTESA;


#ifdef _S_UNICODE
 #define ITTSAttributes          ITTSAttributesW
 #define IID_ITTSAttributes      IID_ITTSAttributesW
 #define PITTSATTRIBUTES         PITTSATTRIBUTESW

#else
 #define ITTSAttributes          ITTSAttributesA
 #define IID_ITTSAttributes      IID_ITTSAttributesA
 #define PITTSATTRIBUTES         PITTSATTRIBUTESA

#endif   // _S_UNICODE



// ITTSBufNotifySink

#undef   INTERFACE
#define  INTERFACE   ITTSBufNotifySink

// {E4963D40-C743-11cd-80E5-00AA003E4B50}
DEFINE_GUID(IID_ITTSBufNotifySink,
0xe4963d40, 0xc743, 0x11cd, 0x80, 0xe5, 0x0, 0xaa, 0x0, 0x3e, 0x4b, 0x50);

DECLARE_INTERFACE_ (ITTSBufNotifySink, IUnknown) {

// IUnknown members

   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

// ITTSBufNotifySink members

   STDMETHOD (TextDataDone)    (THIS_ QWORD, DWORD) PURE;
   STDMETHOD (TextDataStarted) (THIS_ QWORD) PURE;
   STDMETHOD (BookMark)        (THIS_ QWORD, DWORD) PURE;
   STDMETHOD (WordPosition)    (THIS_ QWORD, DWORD) PURE;
   };

typedef ITTSBufNotifySink FAR * PITTSBUFNOTIFYSINK;

// In case anyone uses the W or A interface
#define ITTSBufNotifySinkW          ITTSBufNotifySink
#define IID_ITTSBufNotifySinkW      IID_ITTSBufNotifySink
#define PITTSBUFNOTIFYSINKW         PITTSBUFNOTIFYSINK
#define ITTSBufNotifySinkA          ITTSBufNotifySink
#define IID_ITTSBufNotifySinkA      IID_ITTSBufNotifySink
#define PITTSBUFNOTIFYSINKA         PITTSBUFNOTIFYSINK



/*
 * ITTSCentral
 */

#undef   INTERFACE
#define  INTERFACE   ITTSCentralW

DEFINE_GUID(IID_ITTSCentralW, 0x28016060L, 0x4A47, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSCentralW, IUnknown) {
```

```
    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
    STDMETHOD_(ULONG,Release)  (THIS) PURE;

    // ITTSCentralW members
    STDMETHOD (Inject)         (THIS_ PCWSTR) PURE;
    STDMETHOD (ModeGet)        (THIS_ PTTSMODEINFOW) PURE;
    STDMETHOD (Phoneme)        (THIS_ VOICECHARSET, DWORD, SDATA, PSDATA) PURE;
    STDMETHOD (PosnGet)        (THIS_ PQWORD) PURE;
    STDMETHOD (TextData)       (THIS_ VOICECHARSET, DWORD, SDATA, PVOID, IID) PURE;
    STDMETHOD (ToFileTime)     (THIS_ PQWORD, FILETIME *) PURE;
    STDMETHOD (AudioPause)     (THIS) PURE;
    STDMETHOD (AudioResume)    (THIS) PURE;
    STDMETHOD (AudioReset)     (THIS) PURE;
    STDMETHOD (Register)       (THIS_ PVOID, IID, DWORD*) PURE;
    STDMETHOD (UnRegister)     (THIS_ DWORD) PURE;
    };

typedef ITTSCentralW FAR * PITTSCENTRALW;


#undef   INTERFACE
#define  INTERFACE   ITTSCentralA

DEFINE_GUID(IID_ITTSCentralA, 0x05EB6C6AL, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSCentralA, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
    STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ITTSCentralA members
    STDMETHOD (Inject)         (THIS_ PCSTR) PURE;
    STDMETHOD (ModeGet)        (THIS_ PTTSMODEINFOA) PURE;
    STDMETHOD (Phoneme)        (THIS_ VOICECHARSET, DWORD, SDATA, PSDATA) PURE;
    STDMETHOD (PosnGet)        (THIS_ PQWORD) PURE;
    STDMETHOD (TextData)       (THIS_ VOICECHARSET, DWORD, SDATA, PVOID, IID) PURE;
    STDMETHOD (ToFileTime)     (THIS_ PQWORD, FILETIME *) PURE;
    STDMETHOD (AudioPause)     (THIS) PURE;
    STDMETHOD (AudioResume)    (THIS) PURE;
    STDMETHOD (AudioReset)     (THIS) PURE;
    STDMETHOD (Register)       (THIS_ PVOID, IID, DWORD*) PURE;
    STDMETHOD (UnRegister)     (THIS_ DWORD) PURE;
    };

typedef ITTSCentralA FAR * PITTSCENTRALA;


#ifdef _S_UNICODE
 #define ITTSCentral            ITTSCentralW
 #define IID_ITTSCentral        IID_ITTSCentralW
 #define PITTSCENTRAL           PITTSCENTRALW

#else
 #define ITTSCentral            ITTSCentralA
 #define IID_ITTSCentral        IID_ITTSCentralA
 #define PITTSCENTRAL           PITTSCENTRALA

#endif   // _S_UNICODE



/*
 * ITTSDialogsW
 */

#undef   INTERFACE
```

```
#define  INTERFACE   ITTSDialogsW

DEFINE_GUID(IID_ITTSDialogsW, 0x47F59D00L, 0x4A47, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSDialogsW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ITTSDialogsW members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TranslateDlg)   (THIS_ HWND, PCWSTR) PURE;
   };

typedef ITTSDialogsW FAR * PITTSDIALOGSW;


#undef   INTERFACE
#define  INTERFACE   ITTSDialogsA

DEFINE_GUID(IID_ITTSDialogsA, 0x05EB6C6BL, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSDialogsA, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ITTSDialogsA members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TranslateDlg)   (THIS_ HWND, PCSTR) PURE;
   };

typedef ITTSDialogsA FAR * PITTSDIALOGSA;


#ifdef _S_UNICODE
 #define ITTSDialogs          ITTSDialogsW
 #define IID_ITTSDialogs      IID_ITTSDialogsW
 #define PITTSDIALOGS         PITTSDIALOGSW

#else
 #define ITTSDialogs          ITTSDialogsA
 #define IID_ITTSDialogs      IID_ITTSDialogsA
 #define PITTSDIALOGS         PITTSDIALOGSA

#endif



/*
 * ITTSEnum
 */

#undef   INTERFACE
#define  INTERFACE   ITTSEnumW

DEFINE_GUID(IID_ITTSEnumW, 0x6B837B20L, 0x4A47, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ITTSEnumW, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
```

```
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ITTSEnumW members
    STDMETHOD (Next)            (THIS_ ULONG, PTTSMODEINFOW, ULONG *) PURE;
    STDMETHOD (Skip)            (THIS_ ULONG) PURE;
    STDMETHOD (Reset)           (THIS) PURE;
    STDMETHOD (Clone)           (THIS_ ITTSEnumW * FAR *) PURE;
    STDMETHOD (Select)          (THIS_ GUID, PITTSCENTRALW *, LPUNKNOWN) PURE;
    };

typedef ITTSEnumW FAR * PITTSENUMW;


#undef   INTERFACE
#define  INTERFACE   ITTSEnumA

DEFINE_GUID(IID_ITTSEnumA, 0x05EB6C6DL, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ITTSEnumA, IUnknown) {

    // IUnknown members
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ITTSEnumA members
    STDMETHOD (Next)            (THIS_ ULONG, PTTSMODEINFOA, ULONG *) PURE;
    STDMETHOD (Skip)            (THIS_ ULONG) PURE;
    STDMETHOD (Reset)           (THIS) PURE;
    STDMETHOD (Clone)           (THIS_ ITTSEnumA * FAR *) PURE;
    STDMETHOD (Select)          (THIS_ GUID, PITTSCENTRALA *, LPUNKNOWN) PURE;
    };

typedef ITTSEnumA FAR * PITTSENUMA;


#ifdef _S_UNICODE
 #define ITTSEnum             ITTSEnumW
 #define IID_ITTSEnum         IID_ITTSEnumW
 #define PITTSENUM            PITTSENUMW

#else
 #define ITTSEnum             ITTSEnumA
 #define IID_ITTSEnum         IID_ITTSEnumA
 #define PITTSENUM            PITTSENUMA

#endif



/*
 * ITTSFind
 */

#undef   INTERFACE
#define  INTERFACE   ITTSFindW

DEFINE_GUID(IID_ITTSFindW, 0x7AA42960L, 0x4A47, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ITTSFindW, IUnknown) {

    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // ITTSFindW members
    STDMETHOD (Find)            (THIS_ PTTSMODEINFOW, PTTSMODEINFORANK, PTTSMODEINFOW) PURE;
    STDMETHOD (Select)          (THIS_ GUID, PITTSCENTRALW *, LPUNKNOWN) PURE;
    };
```

```
typedef ITTSFindW FAR * PITTSFINDW;


#undef   INTERFACE
#define  INTERFACE   ITTSFindA

DEFINE_GUID(IID_ITTSFindA, 0x05EB6C6EL, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA, 0x4F);

DECLARE_INTERFACE_ (ITTSFindA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ITTSFindA members
   STDMETHOD (Find)           (THIS_ PTTSMODEINFOA, PTTSMODEINFORANK, PTTSMODEINFOA) PURE;
   STDMETHOD (Select)         (THIS_ GUID, PITTSCENTRALA *, LPUNKNOWN) PURE;
   };

typedef ITTSFindA FAR * PITTSFINDA;


#ifdef _S_UNICODE
 #define ITTSFind             ITTSFindW
 #define IID_ITTSFind         IID_ITTSFindW
 #define PITTSFIND            PITTSFINDW

#else
 #define ITTSFind             ITTSFindA
 #define IID_ITTSFind         IID_ITTSFindA
 #define PITTSFIND            PITTSFINDA

#endif



/*
 * ITTSNotifySink
 */

#undef   INTERFACE
#define  INTERFACE   ITTSNotifySinkW

DEFINE_GUID(IID_ITTSNotifySinkW, 0xC0FA8F40L, 0x4A46, 0x101B, 0x93, 0x1A, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);

DECLARE_INTERFACE_ (ITTSNotifySinkW, IUnknown) {

// IUnknown members

   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

// ITTSNotifySinkW members

   STDMETHOD (AttribChanged)  (THIS_ DWORD) PURE;
   STDMETHOD (AudioStart)     (THIS_ QWORD) PURE;
   STDMETHOD (AudioStop)      (THIS_ QWORD) PURE;
   STDMETHOD (Visual)         (THIS_ QWORD, WCHAR, WCHAR, DWORD, PTTSMOUTH) PURE;
   };

typedef ITTSNotifySinkW FAR * PITTSNOTIFYSINKW;


#undef   INTERFACE
#define  INTERFACE   ITTSNotifySinkA

DEFINE_GUID(IID_ITTSNotifySinkA, 0x05EB6C6FL, 0xDBAB, 0x11CD, 0xB3, 0xCA, 0x00, 0xAA, 0x00, 0x47, 0xBA,
0x4F);
```

```
DECLARE_INTERFACE_ (ITTSNotifySinkA, IUnknown) {

   // IUnknown members
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // ITTSNotifySinkA members
   STDMETHOD (AttribChanged)  (THIS_ DWORD) PURE;
   STDMETHOD (AudioStart)     (THIS_ QWORD) PURE;
   STDMETHOD (AudioStop)      (THIS_ QWORD) PURE;
   STDMETHOD (Visual)         (THIS_ QWORD, CHAR, CHAR, DWORD, PTTSMOUTH) PURE;
   };

typedef ITTSNotifySinkA FAR * PITTSNOTIFYSINKA;


#ifdef _S_UNICODE
 #define ITTSNotifySink       ITTSNotifySinkW
 #define IID_ITTSNotifySink   IID_ITTSNotifySinkW
 #define PITTSNOTIFYSINK      PITTSNOTIFYSINKW

#else
 #define ITTSNotifySink       ITTSNotifySinkA
 #define IID_ITTSNotifySink   IID_ITTSNotifySinkA
 #define PITTSNOTIFYSINK      PITTSNOTIFYSINKA

#endif


/* ITTSNotifySink2 */

#undef   INTERFACE
#define  INTERFACE   ITTSNotifySink2W

// {599F77E1-E42E-11d1-BED8-006008317CE8}
DEFINE_GUID(IID_ITTSNotifySink2W,
0x599f77e1, 0xe42e, 0x11d1, 0xbe, 0xd8, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ITTSNotifySink2W, ITTSNotifySinkW) {

// IUnknown members

   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

// ITTSNotifySink2W members

   STDMETHOD (AttribChanged)  (THIS_ DWORD) PURE;
   STDMETHOD (AudioStart)     (THIS_ QWORD) PURE;
   STDMETHOD (AudioStop)      (THIS_ QWORD) PURE;
   STDMETHOD (Visual)         (THIS_ QWORD, WCHAR, WCHAR, DWORD, PTTSMOUTH) PURE;

   STDMETHOD (Error)          (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Warning)        (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (VisualFuture)   (THIS_ DWORD, QWORD, WCHAR, WCHAR, DWORD, PTTSMOUTH) PURE;
   };

typedef ITTSNotifySink2W FAR * PITTSNOTIFYSINK2W;


#undef   INTERFACE
#define  INTERFACE   ITTSNotifySink2A

// {599F77E2-E42E-11d1-BED8-006008317CE8}
DEFINE_GUID(IID_ITTSNotifySink2A,
0x599f77e2, 0xe42e, 0x11d1, 0xbe, 0xd8, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (ITTSNotifySink2A, ITTSNotifySinkA) {

   // IUnknown members
```

```
    STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
    STDMETHOD_(ULONG,Release)  (THIS) PURE;

    // ITTSNotifySink2A members
    STDMETHOD (AttribChanged)  (THIS_ DWORD) PURE;
    STDMETHOD (AudioStart)     (THIS_ QWORD) PURE;
    STDMETHOD (AudioStop)      (THIS_ QWORD) PURE;
    STDMETHOD (Visual)         (THIS_ QWORD, CHAR, CHAR, DWORD, PTTSMOUTH) PURE;

    STDMETHOD (Error)          (THIS_ LPUNKNOWN) PURE;
    STDMETHOD (Warning)        (THIS_ LPUNKNOWN) PURE;
    STDMETHOD (VisualFuture)   (THIS_ DWORD, QWORD, CHAR, CHAR, DWORD, PTTSMOUTH) PURE;
    };

typedef ITTSNotifySink2A FAR * PITTSNOTIFYSINK2A;


#ifdef _S_UNICODE
 #define ITTSNotifySink2       ITTSNotifySink2W
 #define IID_ITTSNotifySink2   IID_ITTSNotifySink2W
 #define PITTSNOTIFYSINK2      PITTSNOTIFYSINK2W

#else
 #define ITTSNotifySink2       ITTSNotifySink2A
 #define IID_ITTSNotifySink2   IID_ITTSNotifySink2A
 #define PITTSNOTIFYSINK2      PITTSNOTIFYSINK2A

#endif


/*****************************************************************************
High-Level command and control speech recognition API
*/

/*****************************************************************************
defines */


// VCMDNAME member lengths
#define VCMD_APPLEN             ((DWORD)32)
#define VCMD_STATELEN           VCMD_APPLEN
#define VCMD_MICLEN             VCMD_APPLEN
#define VCMD_SPEAKERLEN         VCMD_APPLEN

// dwFlags parameter of IVoiceCmd::MenuCreate
#define  VCMDMC_CREATE_TEMP     0x00000001
#define  VCMDMC_CREATE_NEW      0x00000002
#define  VCMDMC_CREATE_ALWAYS   0x00000004
#define  VCMDMC_OPEN_ALWAYS     0x00000008
#define  VCMDMC_OPEN_EXISTING   0x00000010

// dwFlags parameter of IVoiceCmd::Register
#define  VCMDRF_NOMESSAGES      0
#define  VCMDRF_ALLBUTVUMETER   0x00000001
#define  VCMDRF_VUMETER         0x00000002
#define  VCMDRF_ALLMESSAGES     (VCMDRF_ALLBUTVUMETER | VCMDRF_VUMETER)

// dwFlags parameter of IVoiceCmd::MenuEnum
#define  VCMDEF_DATABASE        0x00000000
#define  VCMDEF_ACTIVE          0x00000001
#define  VCMDEF_SELECTED        0x00000002
#define  VCMDEF_PERMANENT       0x00000004
#define  VCMDEF_TEMPORARY       0x00000008

// dwFlags parameter of IVCmdMenu::Activate
#define  VWGFLAG_ASLEEP         0x00000001

// wPriority parameter of IVCmdMenu::Activate
#define  VCMDACT_NORMAL         (0x8000)
#define  VCMDACT_LOW            (0x4000)
#define  VCMDACT_HIGH           (0xC000)
```

```
// dwFlags of the VCMDCOMMAND structure
#define  VCMDCMD_VERIFY          0x00000001
#define  VCMDCMD_DISABLED_TEMP   0x00000002
#define  VCMDCMD_DISABLED_PERM   0x00000004
#define  VCMDCMD_CANTRENAME      0x00000008

// parameter to any function that processes individual commands
#define  VCMD_BY_POSITION        0x00000001
#define  VCMD_BY_IDENTIFIER      0x00000002


// values for dwAttributes field of IVCmdNotifySink::AttribChanged
#define  IVCNSAC_AUTOGAINENABLE 0x00000001
#define  IVCNSAC_ENABLED         0x00000002
#define  IVCNSAC_AWAKE           0x00000004
#define  IVCNSAC_DEVICE          0x00000008
#define  IVCNSAC_MICROPHONE      0x00000010
#define  IVCNSAC_SPEAKER         0x00000020
#define  IVCNSAC_SRMODE          0x00000040
#define  IVCNSAC_THRESHOLD       0x00000080
#define  IVCNSAC_ORIGINAPP       0x00010000

// values for dwAttributes field of IVTxtNotifySink::AttribChanged
#define  IVTNSAC_DEVICE          0x00000001
#define  IVTNSAC_ENABLED         0x00000002
#define  IVTNSAC_SPEED           0x00000004
#define  IVTNSAC_VOLUME          0x00000008
#define  IVTNSAC_TTSMODE         0x00000010


// values used by IVXxxAttributes::SetMode to set the global speech
// recognition mode
#define  VSRMODE_OFF             0x00000002
#define  VSRMODE_DISABLED        0x00000001
#define  VSRMODE_CMDPAUSED       0x00000004
#define  VSRMODE_CMDONLY         0x00000010
#define  VSRMODE_DCTONLY         0x00000020
#define  VSRMODE_CMDANDDCT       0x00000040


/****************************************************************************
typedefs */

// voice command structure - passed to command menu functions (IVCmdMenu)
typedef struct {
    DWORD   dwSize;         // size of struct including amount of abAction
    DWORD   dwFlags;
    DWORD   dwID;           // Command ID
    DWORD   dwCommand;      // DWORD aligned offset of command string
    DWORD   dwDescription;  // DWORD aligned offset of description string
    DWORD   dwCategory;     // DWORD aligned offset of category string
    DWORD   dwCommandText;  // DWORD aligned offset of command text string
    DWORD   dwAction;       // DWORD aligned offset of action data
    DWORD   dwActionSize;   // size of the action data (could be string or binary)
    BYTE    abData[1];      // command, description, category, and action data
              // (action data is NOT interpreted by voice command)
} VCMDCOMMAND, * PVCMDCOMMAND;


// site information structure - possible parameter to IVoiceCmd::Register
typedef struct {
    DWORD   dwAutoGainEnable;
    DWORD   dwAwakeState;
    DWORD   dwThreshold;
    DWORD   dwDevice;
    DWORD   dwEnable;
    WCHAR   szMicrophone[VCMD_MICLEN];
    WCHAR   szSpeaker[VCMD_SPEAKERLEN];
    GUID    gModeID;
} VCSITEINFOW, *PVCSITEINFOW;
```

```
typedef struct {
    DWORD   dwAutoGainEnable;
    DWORD   dwAwakeState;
    DWORD   dwThreshold;
    DWORD   dwDevice;
    DWORD   dwEnable;
    CHAR    szMicrophone[VCMD_MICLEN];
    CHAR    szSpeaker[VCMD_SPEAKERLEN];
    GUID    gModeID;
} VCSITEINFOA, *PVCSITEINFOA;


// menu name structure
typedef struct {
    WCHAR   szApplication[VCMD_APPLEN]; // unique application name
    WCHAR   szState[VCMD_STATELEN];     // unique application state
} VCMDNAMEW, *PVCMDNAMEW;

typedef struct {
    CHAR    szApplication[VCMD_APPLEN]; // unique application name
    CHAR    szState[VCMD_STATELEN];     // unique application state
} VCMDNAMEA, *PVCMDNAMEA;


#ifdef  _S_UNICODE
 #define VCSITEINFO  VCSITEINFOW
 #define PVCSITEINFO PVCSITEINFOW
 #define VCMDNAME    VCMDNAMEW
 #define PVCMDNAME   PVCMDNAMEW
#else
 #define VCSITEINFO  VCSITEINFOA
 #define PVCSITEINFO PVCSITEINFOA
 #define VCMDNAME    VCMDNAMEA
 #define PVCMDNAME   PVCMDNAMEA
#endif  // _S_UNICODE

/****************************************************************************
interfaces */

/*
 *  IVCmdNotifySink
 */
#undef   INTERFACE
#define  INTERFACE   IVCmdNotifySinkW

DEFINE_GUID(IID_IVCmdNotifySinkW, 0xCCFD7A60L, 0x604D, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVCmdNotifySinkW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVCmdNotifySink members

   STDMETHOD (CommandRecognize) (THIS_ DWORD, PVCMDNAMEW, DWORD, DWORD, PVOID,
                DWORD, PWSTR, PWSTR) PURE;
   STDMETHOD (CommandOther)     (THIS_ PVCMDNAMEW, PWSTR) PURE;
   STDMETHOD (CommandStart)     (THIS) PURE;
   STDMETHOD (MenuActivate)     (THIS_ PVCMDNAMEW, BOOL) PURE;
   STDMETHOD (UtteranceBegin)   (THIS) PURE;
   STDMETHOD (UtteranceEnd)     (THIS) PURE;
   STDMETHOD (VUMeter)          (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)    (THIS_ DWORD) PURE;
   STDMETHOD (Interference)     (THIS_ DWORD) PURE;
};
```

```
typedef IVCmdNotifySinkW FAR * PIVCMDNOTIFYSINKW;


#undef   INTERFACE
#define  INTERFACE   IVCmdNotifySinkA

// {80B25CC0-5540-11b9-C000-5611722E1D15}
DEFINE_GUID(IID_IVCmdNotifySinkA, 0x80b25cc0, 0x5540, 0x11b9, 0xc0, 0x0, 0x56, 0x11, 0x72, 0x2e, 0x1d,
0x15);

DECLARE_INTERFACE_ (IVCmdNotifySinkA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVCmdNotifySinkA members

   STDMETHOD (CommandRecognize) (THIS_ DWORD, PVCMDNAMEA, DWORD, DWORD, PVOID,
                DWORD, PSTR, PSTR) PURE;
   STDMETHOD (CommandOther)     (THIS_ PVCMDNAMEA, PSTR) PURE;
   STDMETHOD (CommandStart)     (THIS) PURE;
   STDMETHOD (MenuActivate)     (THIS_ PVCMDNAMEA, BOOL) PURE;
   STDMETHOD (UtteranceBegin)   (THIS) PURE;
   STDMETHOD (UtteranceEnd)     (THIS) PURE;
   STDMETHOD (VUMeter)          (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)    (THIS_ DWORD) PURE;
   STDMETHOD (Interference)     (THIS_ DWORD) PURE;
};

typedef IVCmdNotifySinkA FAR * PIVCMDNOTIFYSINKA;


#ifdef _S_UNICODE
 #define IVCmdNotifySink        IVCmdNotifySinkW
 #define IID_IVCmdNotifySink    IID_IVCmdNotifySinkW
 #define PIVCMDNOTIFYSINK       PIVCMDNOTIFYSINKW

#else
 #define IVCmdNotifySink        IVCmdNotifySinkA
 #define IID_IVCmdNotifySink    IID_IVCmdNotifySinkA
 #define PIVCMDNOTIFYSINK       PIVCMDNOTIFYSINKA

#endif // _S_UNICODE


/*
 *  IVCmdEnum
 */
#undef   INTERFACE
#define  INTERFACE   IVCmdEnumW

DEFINE_GUID(IID_IVCmdEnumW, 0xD3CC0820L, 0x604D, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVCmdEnumW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVCmdEnum members
   STDMETHOD (Next)            (THIS_ ULONG, PVCMDNAMEW, ULONG *) PURE;
   STDMETHOD (Skip)            (THIS_ ULONG) PURE;
   STDMETHOD (Reset)           (THIS) PURE;
   STDMETHOD (Clone)           (THIS_ IVCmdEnumW * FAR *) PURE;
};
typedef IVCmdEnumW FAR * PIVCMDENUMW;


#undef   INTERFACE
```

```
#define  INTERFACE   IVCmdEnumA

// {E86F9540-DCA2-11CD-A166-00AA004CD65C}
DEFINE_GUID(IID_IVCmdEnumA,
0xE86F9540, 0xDCA2, 0x11CD, 0xA1, 0x66, 0x0, 0xAA, 0x0, 0x4C, 0xD6, 0x5C);

DECLARE_INTERFACE_ (IVCmdEnumA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVCmdEnum members
   STDMETHOD (Next)           (THIS_ ULONG, PVCMDNAMEA, ULONG *) PURE;
   STDMETHOD (Skip)           (THIS_ ULONG) PURE;
   STDMETHOD (Reset)          (THIS) PURE;
   STDMETHOD (Clone)          (THIS_ IVCmdEnumA * FAR *) PURE;
};
typedef IVCmdEnumA FAR * PIVCMDENUMA;


#ifdef _S_UNICODE
 #define IVCmdEnum              IVCmdEnumW
 #define IID_IVCmdEnum          IID_IVCmdEnumW
 #define PIVCMDENUM             PIVCMDENUMW

#else
 #define IVCmdEnum              IVCmdEnumA
 #define IID_IVCmdEnum          IID_IVCmdEnumA
 #define PIVCMDENUM             PIVCMDENUMA

#endif // _S_UNICODE



/*
 *  IEnumSRShare
 */
#undef   INTERFACE
#define  INTERFACE   IEnumSRShareW

// {E97F05C0-81B3-11ce-B763-00AA004CD65C}
DEFINE_GUID(IID_IEnumSRShareW,
0xe97f05c0, 0x81b3, 0x11ce, 0xb7, 0x63, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);

DECLARE_INTERFACE_ (IEnumSRShareW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IEnumSRShare members
   STDMETHOD (Next)           (THIS_ ULONG, PSRSHAREW, ULONG *) PURE;
   STDMETHOD (Skip)           (THIS_ ULONG) PURE;
   STDMETHOD (Reset)          (THIS) PURE;
   STDMETHOD (Clone)          (THIS_ IEnumSRShareW * FAR *) PURE;
   STDMETHOD (New)            (THIS_ DWORD, GUID, PISRCENTRALW *, QWORD *) PURE;
   STDMETHOD (Share)          (THIS_ QWORD, PISRCENTRALW *) PURE;
   STDMETHOD (Detach)         (THIS_ QWORD) PURE;
};
typedef IEnumSRShareW FAR * PIENUMSRSHAREW;


#undef   INTERFACE
#define  INTERFACE   IEnumSRShareA

// {E97F05C1-81B3-11ce-B763-00AA004CD65C}
DEFINE_GUID(IID_IEnumSRShareA,
0xe97f05c1, 0x81b3, 0x11ce, 0xb7, 0x63, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);

DECLARE_INTERFACE_ (IEnumSRShareA, IUnknown) {
```

```
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IEnumSRShare members
    STDMETHOD (Next)            (THIS_ ULONG, PSRSHAREA, ULONG *) PURE;
    STDMETHOD (Skip)            (THIS_ ULONG) PURE;
    STDMETHOD (Reset)           (THIS) PURE;
    STDMETHOD (Clone)           (THIS_ IEnumSRShareA * FAR *) PURE;
    STDMETHOD (New)             (THIS_ DWORD, GUID, PISRCENTRALA *, QWORD *) PURE;
    STDMETHOD (Share)           (THIS_ QWORD, PISRCENTRALA *) PURE;
    STDMETHOD (Detach)          (THIS_ QWORD) PURE;
};
typedef IEnumSRShareA FAR * PIENUMSRSHAREA;


#ifdef _S_UNICODE
 #define IEnumSRShare               IEnumSRShareW
 #define IID_IEnumSRShare           IID_IEnumSRShareW
 #define PIENUMSRSHARE              PIENUMSRSHAREW

#else
 #define IEnumSRShare               IEnumSRShareA
 #define IID_IEnumSRShare           IID_IEnumSRShareA
 #define PIENUMSRSHARE              PIENUMSRSHAREA

#endif // _S_UNICODE



/*
 *  IVCmdMenu
 */
#undef   INTERFACE
#define  INTERFACE   IVCmdMenuW

DEFINE_GUID(IID_IVCmdMenuW, 0xDAC54F60L, 0x604D, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVCmdMenuW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVCmdMenu members
    STDMETHOD (Activate)        (THIS_ HWND, DWORD) PURE;
    STDMETHOD (Deactivate)      (THIS) PURE;
    STDMETHOD (TrainMenuDlg)    (THIS_ HWND, PCWSTR) PURE;
    STDMETHOD (Num)             (THIS_ DWORD *) PURE;
    STDMETHOD (Get)             (THIS_ DWORD, DWORD, DWORD, PSDATA, DWORD *) PURE;
    STDMETHOD (Set)             (THIS_ DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD (Add)             (THIS_ DWORD, SDATA, DWORD *) PURE;
    STDMETHOD (Remove)          (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (ListGet)         (THIS_ PCWSTR, PSDATA, DWORD *) PURE;
    STDMETHOD (ListSet)         (THIS_ PCWSTR, DWORD, SDATA) PURE;
    STDMETHOD (EnableItem)      (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (SetItem)         (THIS_ DWORD, DWORD, DWORD) PURE;
};

typedef IVCmdMenuW FAR * PIVCMDMENUW;


#undef   INTERFACE
#define  INTERFACE   IVCmdMenuA

// {746141E0-5543-11b9-C000-5611722E1D15}
DEFINE_GUID(IID_IVCmdMenuA, 0x746141e0, 0x5543, 0x11b9, 0xc0, 0x0, 0x56, 0x11, 0x72, 0x2e, 0x1d, 0x15);

DECLARE_INTERFACE_ (IVCmdMenuA, IUnknown) {
```

```
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVCmdMenu members
    STDMETHOD (Activate)        (THIS_ HWND, DWORD) PURE;
    STDMETHOD (Deactivate)      (THIS) PURE;
    STDMETHOD (TrainMenuDlg)    (THIS_ HWND, PCSTR) PURE;
    STDMETHOD (Num)             (THIS_ DWORD *) PURE;
    STDMETHOD (Get)             (THIS_ DWORD, DWORD, DWORD, PSDATA, DWORD *) PURE;
    STDMETHOD (Set)             (THIS_ DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD (Add)             (THIS_ DWORD, SDATA, DWORD *) PURE;
    STDMETHOD (Remove)          (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (ListGet)         (THIS_ PCSTR, PSDATA, DWORD *) PURE;
    STDMETHOD (ListSet)         (THIS_ PCSTR, DWORD, SDATA) PURE;
    STDMETHOD (EnableItem)      (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (SetItem)         (THIS_ DWORD, DWORD, DWORD) PURE;
};

typedef IVCmdMenuA FAR * PIVCMDMENUA;


#ifdef _S_UNICODE
 #define IVCmdMenu      IVCmdMenuW
 #define IID_IVCmdMenu  IID_IVCmdMenuW
 #define PIVCMDMENU     PIVCMDMENUW

#else
 #define IVCmdMenu      IVCmdMenuA
 #define IID_IVCmdMenu  IID_IVCmdMenuA
 #define PIVCMDMENU     PIVCMDMENUA

#endif // _S_UNICODE



/*
 *  IVoiceCmd
 */
#undef   INTERFACE
#define  INTERFACE   IVoiceCmdW

DEFINE_GUID(IID_IVoiceCmdW, 0xE0DCC220L, 0x604D, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVoiceCmdW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVoiceCmd members
    STDMETHOD (Register)        (THIS_ PCWSTR, LPUNKNOWN, GUID, DWORD,
                    PVCSITEINFOW) PURE;
    STDMETHOD (MenuEnum)        (THIS_ DWORD, PCWSTR, PCWSTR, PIVCMDENUMW *) PURE;
    STDMETHOD (MenuCreate)      (THIS_ PVCMDNAMEW, PLANGUAGEW, DWORD,
                    PIVCMDMENUW *) PURE;
    STDMETHOD (MenuDelete)      (THIS_ PVCMDNAMEW) PURE;
    STDMETHOD (CmdMimic)        (THIS_ PVCMDNAMEW, PCWSTR) PURE;
};

typedef IVoiceCmdW FAR * PIVOICECMDW;


#undef   INTERFACE
#define  INTERFACE   IVoiceCmdA

// {C63A2B30-5543-11b9-C000-5611722E1D15}
DEFINE_GUID(IID_IVoiceCmdA, 0xc63a2b30, 0x5543, 0x11b9, 0xc0, 0x0, 0x56, 0x11, 0x72, 0x2e, 0x1d, 0x15);

DECLARE_INTERFACE_ (IVoiceCmdA, IUnknown) {
```

```
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVoiceCmd members
    STDMETHOD (Register)        (THIS_ PCSTR, LPUNKNOWN, GUID, DWORD,
                        PVCSITEINFOA) PURE;
    STDMETHOD (MenuEnum)        (THIS_ DWORD, PCSTR, PCSTR, PIVCMDENUMA *) PURE;
    STDMETHOD (MenuCreate)      (THIS_ PVCMDNAMEA, PLANGUAGEA, DWORD,
                        PIVCMDMENUA *) PURE;
    STDMETHOD (MenuDelete)      (THIS_ PVCMDNAMEA) PURE;
    STDMETHOD (CmdMimic)        (THIS_ PVCMDNAMEA, PCSTR) PURE;
};

typedef IVoiceCmdA FAR * PIVOICECMDA;


#ifdef _S_UNICODE
 #define IVoiceCmd      IVoiceCmdW
 #define IID_IVoiceCmd  IID_IVoiceCmdW
 #define PIVOICECMD     PIVOICECMDW

#else
 #define IVoiceCmd      IVoiceCmdA
 #define IID_IVoiceCmd  IID_IVoiceCmdA
 #define PIVOICECMD     PIVOICECMDA

#endif //_S_UNICODE


/*
 *  IVCmdAttributes
 */
#undef   INTERFACE
#define  INTERFACE   IVCmdAttributesW

DEFINE_GUID(IID_IVCmdAttributesW, 0xE5F24680L, 0x6053, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVCmdAttributesW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVCmdAttributes members
    STDMETHOD (AutoGainEnableGet) (THIS_ DWORD *) PURE;
    STDMETHOD (AutoGainEnableSet) (THIS_ DWORD) PURE;
    STDMETHOD (AwakeStateGet)     (THIS_ DWORD *) PURE;
    STDMETHOD (AwakeStateSet)     (THIS_ DWORD) PURE;
    STDMETHOD (ThresholdGet)      (THIS_ DWORD *) PURE;
    STDMETHOD (ThresholdSet)      (THIS_ DWORD) PURE;
    STDMETHOD (DeviceGet)         (THIS_ DWORD *) PURE;
    STDMETHOD (DeviceSet)         (THIS_ DWORD) PURE;
    STDMETHOD (EnabledGet)        (THIS_ DWORD *) PURE;
    STDMETHOD (EnabledSet)        (THIS_ DWORD) PURE;
    STDMETHOD (MicrophoneGet)     (THIS_ PWSTR, DWORD, DWORD *) PURE;
    STDMETHOD (MicrophoneSet)     (THIS_ PCWSTR) PURE;
    STDMETHOD (SpeakerGet)        (THIS_ PWSTR, DWORD, DWORD *) PURE;
    STDMETHOD (SpeakerSet)        (THIS_ PCWSTR) PURE;
    STDMETHOD (SRModeGet)         (THIS_ GUID *) PURE;
    STDMETHOD (SRModeSet)         (THIS_ GUID) PURE;
};

typedef IVCmdAttributesW FAR * PIVCMDATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   IVCmdAttributesA

// {FFF5DF80-5544-11b9-C000-5611722E1D15}
```

```
DEFINE_GUID(IID_IVCmdAttributesA, 0xfff5df80, 0x5544, 0x11b9, 0xc0, 0x0, 0x56, 0x11, 0x72, 0x2e, 0x1d,
0x15);

DECLARE_INTERFACE_ (IVCmdAttributesA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVCmdAttributes members
   STDMETHOD (AutoGainEnableGet) (THIS_ DWORD *) PURE;
   STDMETHOD (AutoGainEnableSet) (THIS_ DWORD) PURE;
   STDMETHOD (AwakeStateGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (AwakeStateSet)     (THIS_ DWORD) PURE;
   STDMETHOD (ThresholdGet)      (THIS_ DWORD *) PURE;
   STDMETHOD (ThresholdSet)      (THIS_ DWORD) PURE;
   STDMETHOD (DeviceGet)         (THIS_ DWORD *) PURE;
   STDMETHOD (DeviceSet)         (THIS_ DWORD) PURE;
   STDMETHOD (EnabledGet)        (THIS_ DWORD *) PURE;
   STDMETHOD (EnabledSet)        (THIS_ DWORD) PURE;
   STDMETHOD (MicrophoneGet)     (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (MicrophoneSet)     (THIS_ PCSTR) PURE;
   STDMETHOD (SpeakerGet)        (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD (SpeakerSet)        (THIS_ PCSTR) PURE;
   STDMETHOD (SRModeGet)         (THIS_ GUID *) PURE;
   STDMETHOD (SRModeSet)         (THIS_ GUID) PURE;
};

typedef IVCmdAttributesA FAR * PIVCMDATTRIBUTESA;


#ifdef _S_UNICODE
 #define IVCmdAttributes        IVCmdAttributesW
 #define IID_IVCmdAttributes    IID_IVCmdAttributesW
 #define PIVCMDATTRIBUTES       PIVCMDATTRIBUTESW

#else
 #define IVCmdAttributes        IVCmdAttributesA
 #define IID_IVCmdAttributes    IID_IVCmdAttributesA
 #define PIVCMDATTRIBUTES       PIVCMDATTRIBUTESA

#endif // _S_UNICODE



/*
 *  IVCmdDialog
 */
#undef   INTERFACE
#define  INTERFACE   IVCmdDialogsW

DEFINE_GUID(IID_IVCmdDialogsW, 0xEE39B8A0L, 0x6053, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC,
0x2C);

DECLARE_INTERFACE_ (IVCmdDialogsW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // IVCmdDialogs members
   STDMETHOD (AboutDlg)          (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (GeneralDlg)        (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (LexiconDlg)        (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainGeneralDlg)   (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TrainMicDlg)       (THIS_ HWND, PCWSTR) PURE;
};

typedef IVCmdDialogsW FAR * PIVCMDDIALOGSW;


#undef   INTERFACE
```

```
#define  INTERFACE   IVCmdDialogsA

// {AA8FE260-5545-11b9-C000-5611722E1D15}
DEFINE_GUID(IID_IVCmdDialogsA, 0xaa8fe260, 0x5545, 0x11b9, 0xc0, 0x0, 0x56, 0x11, 0x72, 0x2e, 0x1d,
0x15);

DECLARE_INTERFACE_ (IVCmdDialogsA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // IVCmdDialogs members
   STDMETHOD (AboutDlg)          (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)        (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)        (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainGeneralDlg)   (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainMicDlg)       (THIS_ HWND, PCSTR) PURE;
};

typedef IVCmdDialogsA FAR * PIVCMDDIALOGSA;


#ifdef _S_UNICODE
 #define IVCmdDialogs       IVCmdDialogsW
 #define IID_IVCmdDialogs   IID_IVCmdDialogsW
 #define PIVCMDDIALOGS      PIVCMDDIALOGSW

#else
 #define IVCmdDialogs       IVCmdDialogsA
 #define IID_IVCmdDialogs   IID_IVCmdDialogsA
 #define PIVCMDDIALOGS      PIVCMDDIALOGSA

#endif // _S_UNICODE


/****************************************************************************
class guids */

// DEFINE_GUID(CLSID_VCmd, 0x93898800L, 0x604D, 0x101B, 0x99, 0x26, 0x00, 0xAA, 0x00, 0x3C, 0xFC, 0x2C);
// {6D40D820-0BA7-11ce-A166-00AA004CD65C}
DEFINE_GUID(CLSID_VCmd,
0x6d40d820, 0xba7, 0x11ce, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);
// {89F70C30-8636-11ce-B763-00AA004CD65C}
DEFINE_GUID(CLSID_SRShare,
0x89f70c30, 0x8636, 0x11ce, 0xb7, 0x63, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);



/****************************************************************************
High-Level dictation speech recognition API
*/

/****************************************************************************
defines */
#define     VDCT_TOPICNAMELEN       32

// bit flags for the dwReason field of IVDctNotifySink::TextChanged
#define     VDCT_TEXTADDED          0x00000001
#define     VDCT_TEXTREMOVED        0x00000002
#define     VDCT_TEXTREPLACED       0x00000004

// bit flags for the dwReason field of IVDctText::TextRemove/TextSet
#define     VDCT_TEXTCLEAN          0x00010000
#define     VDCT_TEXTKEEPRESULTS    0x00020000

// bit flags for dwFlags of IVDctGUI::FlagsSet
#define     VDCTGUIF_VISIBLE        0x00000001
#define     VDCTGUIF_DONTMOVE       0x00000002
#define     VDCTGUIF_ADDWORD        0x00000004
```

```
// bit flags for dwFX of IVDctText::FX
#define     VDCTFX_CAPFIRST         0x00000001
#define     VDCTFX_LOWERFIRST       0x00000002
#define     VDCTFX_TOGGLEFIRST      0x00000003
#define     VDCTFX_CAPALL           0x00000004
#define     VDCTFX_LOWERALL         0x00000005
#define     VDCTFX_REMOVESPACES     0x00000006
#define     VDCTFX_KEEPONLYFIRSTLETTER 0x00000007

/*****************************************************************************
typedefs */

// site information structure - used for IVoiceDictation::SiteInfoGet/Set
typedef struct {
    DWORD   dwAutoGainEnable;
    DWORD   dwAwakeState;
    DWORD   dwThreshold;
    DWORD   dwDevice;
    DWORD   dwEnable;
    WCHAR   szMicrophone[VCMD_MICLEN];
    WCHAR   szSpeaker[VCMD_SPEAKERLEN];
    GUID    gModeID;
} VDSITEINFOW, *PVDSITEINFOW;

typedef struct {
    DWORD   dwAutoGainEnable;
    DWORD   dwAwakeState;
    DWORD   dwThreshold;
    DWORD   dwDevice;
    DWORD   dwEnable;
    CHAR    szMicrophone[VCMD_MICLEN];
    CHAR    szSpeaker[VCMD_SPEAKERLEN];
    GUID    gModeID;
} VDSITEINFOA, *PVDSITEINFOA;


// topic structure used by the IVoiceDictation object
typedef struct {
    WCHAR       szTopic[VDCT_TOPICNAMELEN];
    LANGUAGEW   language;
} VDCTTOPICW, *PVDCTTOPICW;

typedef struct {
    CHAR        szTopic[VDCT_TOPICNAMELEN];
    LANGUAGEA   language;
} VDCTTOPICA, *PVDCTTOPICA;


#ifdef  _S_UNICODE
 #define VDSITEINFO  VDSITEINFOW
 #define PVDSITEINFO PVDSITEINFOW
 #define VDCTTOPIC   VDCTTOPICW
 #define PVDCTTOPIC  PVDCTTOPICW
#else
 #define VDSITEINFO  VDSITEINFOA
 #define PVDSITEINFO PVDSITEINFOA
 #define VDCTTOPIC   VDCTTOPICA
 #define PVDCTTOPIC  PVDCTTOPICA
#endif  // _S_UNICODE


// memory maintenance structure used by MemoryGet/Set in IVDctAttributes
typedef struct {
    DWORD   dwMaxRAM;
    DWORD   dwMaxTime;
    DWORD   dwMaxWords;
    BOOL    fKeepAudio;
    BOOL    fKeepCorrection;
    BOOL    fKeepEval;
} VDCTMEMORY, *PVDCTMEMORY;
```

```
// bookmark definition
typedef struct {
    DWORD   dwID;
    DWORD   dwPosn;
} VDCTBOOKMARK, *PVDCTBOOKMARK;



/****************************************************************************
interfaces */

/*
 *  IVDctNotifySink
 */
#undef   INTERFACE
#define  INTERFACE   IVDctNotifySinkW

// {5FEB8800-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctNotifySinkW,
0x5feb8800, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctNotifySinkW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctNotifySink members
   STDMETHOD (CommandBuiltIn)        (THIS_ PWSTR) PURE;
   STDMETHOD (CommandOther)          (THIS_ PWSTR) PURE;
   STDMETHOD (CommandRecognize)      (THIS_ DWORD, DWORD, DWORD, PVOID, PWSTR) PURE;
   STDMETHOD (TextSelChanged)        (THIS) PURE;
   STDMETHOD (TextChanged)           (THIS_ DWORD) PURE;
   STDMETHOD (TextBookmarkChanged)   (THIS_ DWORD) PURE;
   STDMETHOD (PhraseStart)           (THIS) PURE;
   STDMETHOD (PhraseFinish)          (THIS_ DWORD, PSRPHRASEW) PURE;
   STDMETHOD (PhraseHypothesis)      (THIS_ DWORD, PSRPHRASEW) PURE;
   STDMETHOD (UtteranceBegin)        (THIS) PURE;
   STDMETHOD (UtteranceEnd)          (THIS) PURE;
   STDMETHOD (VUMeter)               (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)         (THIS_ DWORD) PURE;
   STDMETHOD (Interference)          (THIS_ DWORD) PURE;
   STDMETHOD (Training)              (THIS_ DWORD) PURE;
   STDMETHOD (Dictating)             (THIS_ PCWSTR, BOOL) PURE;
};

typedef IVDctNotifySinkW FAR * PIVDCTNOTIFYSINKW;


#undef   INTERFACE
#define  INTERFACE   IVDctNotifySinkA

// {88AD7DC0-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctNotifySinkA,
0x88ad7dc0, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctNotifySinkA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctNotifySinkA members
   STDMETHOD (CommandBuiltIn)        (THIS_ PSTR) PURE;
   STDMETHOD (CommandOther)          (THIS_ PSTR) PURE;
   STDMETHOD (CommandRecognize)      (THIS_ DWORD, DWORD, DWORD, PVOID, PSTR) PURE;
   STDMETHOD (TextSelChanged)        (THIS) PURE;
   STDMETHOD (TextChanged)           (THIS_ DWORD) PURE;
   STDMETHOD (TextBookmarkChanged)   (THIS_ DWORD) PURE;
   STDMETHOD (PhraseStart)           (THIS) PURE;
```

```
   STDMETHOD (PhraseFinish)          (THIS_ DWORD, PSRPHRASEA) PURE;
   STDMETHOD (PhraseHypothesis)      (THIS_ DWORD, PSRPHRASEA) PURE;
   STDMETHOD (UtteranceBegin)        (THIS) PURE;
   STDMETHOD (UtteranceEnd)          (THIS) PURE;
   STDMETHOD (VUMeter)               (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)         (THIS_ DWORD) PURE;
   STDMETHOD (Interference)          (THIS_ DWORD) PURE;
   STDMETHOD (Training)              (THIS_ DWORD) PURE;
   STDMETHOD (Dictating)             (THIS_ PCSTR, BOOL) PURE;
};

typedef IVDctNotifySinkA FAR * PIVDCTNOTIFYSINKA;


#ifdef _S_UNICODE
 #define IVDctNotifySink         IVDctNotifySinkW
 #define IID_IVDctNotifySink     IID_IVDctNotifySinkW
 #define PIVDCTNOTIFYSINK        PIVDCTNOTIFYSINKW

#else
 #define IVDctNotifySink         IVDctNotifySinkA
 #define IID_IVDctNotifySink     IID_IVDctNotifySinkA
 #define PIVDCTNOTIFYSINK        PIVDCTNOTIFYSINKA

#endif // _S_UNICODE



/*
 *  IVDctNotifySink2
 */
#undef   INTERFACE
#define  INTERFACE   IVDctNotifySink2W

// {599F77E3-E42E-11d1-BED8-006008317CE8}
DEFINE_GUID(IID_IVDctNotifySink2W,
0x599f77e3, 0xe42e, 0x11d1, 0xbe, 0xd8, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IVDctNotifySink2W, IVDctNotifySinkW) {

   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVDctNotifySink2 members
   STDMETHOD (CommandBuiltIn)        (THIS_ PWSTR) PURE;
   STDMETHOD (CommandOther)          (THIS_ PWSTR) PURE;
   STDMETHOD (CommandRecognize)      (THIS_ DWORD, DWORD, DWORD, PVOID, PWSTR) PURE;
   STDMETHOD (TextSelChanged)        (THIS) PURE;
   STDMETHOD (TextChanged)           (THIS_ DWORD) PURE;
   STDMETHOD (TextBookmarkChanged)   (THIS_ DWORD) PURE;
   STDMETHOD (PhraseStart)           (THIS) PURE;
   STDMETHOD (PhraseFinish)          (THIS_ DWORD, PSRPHRASEW) PURE;
   STDMETHOD (PhraseHypothesis)      (THIS_ DWORD, PSRPHRASEW) PURE;
   STDMETHOD (UtteranceBegin)        (THIS) PURE;
   STDMETHOD (UtteranceEnd)          (THIS) PURE;
   STDMETHOD (VUMeter)               (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)         (THIS_ DWORD) PURE;
   STDMETHOD (Interference)          (THIS_ DWORD) PURE;
   STDMETHOD (Training)              (THIS_ DWORD) PURE;
   STDMETHOD (Dictating)             (THIS_ PCWSTR, BOOL) PURE;

   STDMETHOD (Error)           (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Warning)         (THIS_ LPUNKNOWN) PURE;
};

typedef IVDctNotifySink2W FAR * PIVDCTNOTIFYSINK2W;


#undef   INTERFACE
```

```
#define  INTERFACE   IVDctNotifySink2A

// {599F77E4-E42E-11d1-BED8-006008317CE8}
DEFINE_GUID(IID_IVDctNotifySink2A,
0x599f77e4, 0xe42e, 0x11d1, 0xbe, 0xd8, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IVDctNotifySink2A, IVDctNotifySinkA) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctNotifySink2A members
   STDMETHOD (CommandBuiltIn)        (THIS_ PSTR) PURE;
   STDMETHOD (CommandOther)          (THIS_ PSTR) PURE;
   STDMETHOD (CommandRecognize)      (THIS_ DWORD, DWORD, DWORD, PVOID, PSTR) PURE;
   STDMETHOD (TextSelChanged)        (THIS) PURE;
   STDMETHOD (TextChanged)           (THIS_ DWORD) PURE;
   STDMETHOD (TextBookmarkChanged)   (THIS_ DWORD) PURE;
   STDMETHOD (PhraseStart)           (THIS) PURE;
   STDMETHOD (PhraseFinish)          (THIS_ DWORD, PSRPHRASEA) PURE;
   STDMETHOD (PhraseHypothesis)      (THIS_ DWORD, PSRPHRASEA) PURE;
   STDMETHOD (UtteranceBegin)        (THIS) PURE;
   STDMETHOD (UtteranceEnd)          (THIS) PURE;
   STDMETHOD (VUMeter)               (THIS_ WORD) PURE;
   STDMETHOD (AttribChanged)         (THIS_ DWORD) PURE;
   STDMETHOD (Interference)          (THIS_ DWORD) PURE;
   STDMETHOD (Training)              (THIS_ DWORD) PURE;
   STDMETHOD (Dictating)             (THIS_ PCSTR, BOOL) PURE;

   STDMETHOD (Error)           (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (Warning)         (THIS_ LPUNKNOWN) PURE;
};

typedef IVDctNotifySink2A FAR * PIVDCTNOTIFYSINK2A;


#ifdef _S_UNICODE
 #define IVDctNotifySink2         IVDctNotifySink2W
 #define IID_IVDctNotifySink2     IID_IVDctNotifySink2W
 #define PIVDCTNOTIFYSINK2        PIVDCTNOTIFYSINK2W

#else
 #define IVDctNotifySink2         IVDctNotifySink2A
 #define IID_IVDctNotifySink2     IID_IVDctNotifySink2A
 #define PIVDCTNOTIFYSINK2        PIVDCTNOTIFYSINK2A

#endif // _S_UNICODE



/*
 *  IVoiceDictation
 */
#undef   INTERFACE
#define  INTERFACE   IVoiceDictationW

// {88AD7DC3-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVoiceDictationW,
0x88ad7dc3, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVoiceDictationW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVoiceDictation members
   STDMETHOD (Register)          (THIS_ PCWSTR, PCWSTR, LPSTORAGE, PCWSTR,
                                  PIVDCTNOTIFYSINK, GUID, DWORD) PURE;
   STDMETHOD (SiteInfoGet)       (THIS_ PCWSTR, PVDSITEINFOW) PURE;
```

```
    STDMETHOD (SiteInfoSet)       (THIS_ PCWSTR, PVDSITEINFOW) PURE;
    STDMETHOD (SessionSerialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (SessionDeserialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (TopicEnum)         (THIS_ PSDATA) PURE;
    STDMETHOD (TopicAddString)    (THIS_ PCWSTR, LANGUAGEW *) PURE;
    STDMETHOD (TopicRemove)       (THIS_ PCWSTR) PURE;
    STDMETHOD (TopicSerialize)    (THIS_ LPSTORAGE) PURE;
    STDMETHOD (TopicDeserialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (Activate)          (THIS_ HWND) PURE;
    STDMETHOD (Deactivate)        (THIS) PURE;
    STDMETHOD (TopicAddGrammar)   (THIS_ PCWSTR, SDATA) PURE;
    STDMETHOD (CFGSet)            (THIS_ LANGID, PCWSTR, PCWSTR) PURE;
};

typedef IVoiceDictationW FAR * PIVOICEDICTATIONW;


#undef   INTERFACE
#define  INTERFACE   IVoiceDictationA

// {88AD7DC4-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVoiceDictationA,
0x88ad7dc4, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVoiceDictationA, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVoiceDictation members
    STDMETHOD (Register)          (THIS_ PCSTR, PCSTR, LPSTORAGE, PCSTR,
                PIVDCTNOTIFYSINK, GUID, DWORD) PURE;
    STDMETHOD (SiteInfoGet)       (THIS_ PCSTR, PVDSITEINFOA) PURE;
    STDMETHOD (SiteInfoSet)       (THIS_ PCSTR, PVDSITEINFOA) PURE;
    STDMETHOD (SessionSerialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (SessionDeserialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (TopicEnum)         (THIS_ PSDATA) PURE;
    STDMETHOD (TopicAddString)    (THIS_ PCSTR, LANGUAGEA *) PURE;
    STDMETHOD (TopicRemove)       (THIS_ PCSTR) PURE;
    STDMETHOD (TopicSerialize)    (THIS_ LPSTORAGE) PURE;
    STDMETHOD (TopicDeserialize) (THIS_ LPSTORAGE) PURE;
    STDMETHOD (Activate)          (THIS_ HWND) PURE;
    STDMETHOD (Deactivate)        (THIS) PURE;
    STDMETHOD (TopicAddGrammar)   (THIS_ PCSTR, SDATA) PURE;
    STDMETHOD (CFGSet)            (THIS_ LANGID, PCSTR, PCSTR) PURE;
};

typedef IVoiceDictationA FAR * PIVOICEDICTATIONA;


#ifdef _S_UNICODE
 #define IVoiceDictation     IVoiceDictationW
 #define IID_IVoiceDictation IID_IVoiceDictationW
 #define PIVOICEDICTATION    PIVOICEDICTATIONW

#else
 #define IVoiceDictation     IVoiceDictationA
 #define IID_IVoiceDictation IID_IVoiceDictationA
 #define PIVOICEDICTATION    PIVOICEDICTATIONA

#endif //_S_UNICODE



/*
 *  IVDctText
 */
#undef   INTERFACE
#define  INTERFACE   IVDctTextW

// {6D62B3A0-6893-11cf-9B8B-08005AFC3A41}
```

```
DEFINE_GUID(IID_IVDctTextW,
0x6d62b3a0, 0x6893, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctTextW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctText members
   STDMETHOD (Lock)              (THIS) PURE;
   STDMETHOD (UnLock)            (THIS) PURE;
   STDMETHOD (TextGet)           (THIS_ DWORD, DWORD, PSDATA) PURE;
   STDMETHOD (TextSet)           (THIS_ DWORD, DWORD, PCWSTR, DWORD) PURE;
   STDMETHOD (TextMove)          (THIS_ DWORD, DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextRemove)        (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextSelSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (TextSelGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TextLengthGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (GetChanges)        (THIS_ DWORD *, DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (BookmarkAdd)       (THIS_ PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkRemove)    (THIS_ DWORD) PURE;
   STDMETHOD (BookmarkMove)      (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (BookmarkQuery)     (THIS_ DWORD, PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkEnum)      (THIS_ DWORD, DWORD, PVDCTBOOKMARK *,
                                 DWORD *) PURE;
   STDMETHOD (Hint)              (THIS_ PCWSTR) PURE;
   STDMETHOD (Words)             (THIS_ PCWSTR) PURE;
   STDMETHOD (ResultsGet)        (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                                 LPUNKNOWN *) PURE;
   STDMETHOD (FX)                (THIS_ DWORD) PURE;
};
typedef IVDctTextW FAR * PIVDCTTEXTW;


#undef   INTERFACE
#define  INTERFACE   IVDctTextA

// {6D62B3A1-6893-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctTextA,
0x6d62b3a1, 0x6893, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctTextA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctText members
   STDMETHOD (Lock)              (THIS) PURE;
   STDMETHOD (UnLock)            (THIS) PURE;
   STDMETHOD (TextGet)           (THIS_ DWORD, DWORD, PSDATA) PURE;
   STDMETHOD (TextSet)           (THIS_ DWORD, DWORD, PCSTR, DWORD) PURE;
   STDMETHOD (TextMove)          (THIS_ DWORD, DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextRemove)        (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextSelSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (TextSelGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TextLengthGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (GetChanges)        (THIS_ DWORD *, DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (BookmarkAdd)       (THIS_ PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkRemove)    (THIS_ DWORD) PURE;
   STDMETHOD (BookmarkMove)      (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (BookmarkQuery)     (THIS_ DWORD, PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkEnum)      (THIS_ DWORD, DWORD, PVDCTBOOKMARK *,
                        DWORD *) PURE;
   STDMETHOD (Hint)              (THIS_ PCSTR) PURE;
   STDMETHOD (Words)             (THIS_ PCSTR) PURE;
   STDMETHOD (ResultsGet)        (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                        LPUNKNOWN *) PURE;
   STDMETHOD (FX)                (THIS_ DWORD) PURE;
};
typedef IVDctTextA FAR * PIVDCTTEXTA;
```

```
#ifdef _S_UNICODE
 #define IVDctText       IVDctTextW
 #define IID_IVDctText   IID_IVDctTextW
 #define PIVDCTTEXT      PIVDCTTEXTW

#else
 #define IVDctText       IVDctTextA
 #define IID_IVDctText   IID_IVDctTextA
 #define PIVDCTTEXT      PIVDCTTEXTA

#endif //_S_UNICODE


/*
 *  IVDctText2
 */
#undef   INTERFACE
#define  INTERFACE   IVDctText2W

// {238004E1-F0C4-11d1-BED9-006008317CE8}
DEFINE_GUID(IID_IVDctText2W,
0x238004e1, 0xf0c4, 0x11d1, 0xbe, 0xd9, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IVDctText2W, IVDctTextW) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctText members
   STDMETHOD (Lock)              (THIS) PURE;
   STDMETHOD (UnLock)            (THIS) PURE;
   STDMETHOD (TextGet)           (THIS_ DWORD, DWORD, PSDATA) PURE;
   STDMETHOD (TextSet)           (THIS_ DWORD, DWORD, PCWSTR, DWORD) PURE;
   STDMETHOD (TextMove)          (THIS_ DWORD, DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextRemove)        (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextSelSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (TextSelGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TextLengthGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (GetChanges)        (THIS_ DWORD *, DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (BookmarkAdd)       (THIS_ PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkRemove)    (THIS_ DWORD) PURE;
   STDMETHOD (BookmarkMove)      (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (BookmarkQuery)     (THIS_ DWORD, PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkEnum)      (THIS_ DWORD, DWORD, PVDCTBOOKMARK *,
                                 DWORD *) PURE;
   STDMETHOD (Hint)              (THIS_ PCWSTR) PURE;
   STDMETHOD (Words)             (THIS_ PCWSTR) PURE;
   STDMETHOD (ResultsGet)        (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                                 LPUNKNOWN *) PURE;
   STDMETHOD (FX)                (THIS_ DWORD) PURE;

   // IVDctText2
   STDMETHOD (ITNApply)          (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ITNExpand)         (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ResultsGet2)       (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                                     LPUNKNOWN *, DWORD*, DWORD*) PURE;
   STDMETHOD (ResultsSet)        (THIS_ DWORD, DWORD,
                                     LPUNKNOWN, DWORD, DWORD) PURE;

};
typedef IVDctText2W FAR * PIVDCTTEXT2W;


#undef   INTERFACE
#define  INTERFACE   IVDctText2A

// {238004E2-F0C4-11d1-BED9-006008317CE8}
DEFINE_GUID(IID_IVDctText2A,
```

```
0x238004e2, 0xf0c4, 0x11d1, 0xbe, 0xd9, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IVDctText2A, IVDctTextA) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctText members
   STDMETHOD (Lock)              (THIS) PURE;
   STDMETHOD (UnLock)            (THIS) PURE;
   STDMETHOD (TextGet)           (THIS_ DWORD, DWORD, PSDATA) PURE;
   STDMETHOD (TextSet)           (THIS_ DWORD, DWORD, PCSTR, DWORD) PURE;
   STDMETHOD (TextMove)          (THIS_ DWORD, DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextRemove)        (THIS_ DWORD, DWORD, DWORD) PURE;
   STDMETHOD (TextSelSet)        (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (TextSelGet)        (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD (TextLengthGet)     (THIS_ DWORD *) PURE;
   STDMETHOD (GetChanges)        (THIS_ DWORD *, DWORD *, DWORD *, DWORD *) PURE;
   STDMETHOD (BookmarkAdd)       (THIS_ PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkRemove)    (THIS_ DWORD) PURE;
   STDMETHOD (BookmarkMove)      (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (BookmarkQuery)     (THIS_ DWORD, PVDCTBOOKMARK) PURE;
   STDMETHOD (BookmarkEnum)      (THIS_ DWORD, DWORD, PVDCTBOOKMARK *,
                      DWORD *) PURE;
   STDMETHOD (Hint)              (THIS_ PCSTR) PURE;
   STDMETHOD (Words)             (THIS_ PCSTR) PURE;
   STDMETHOD (ResultsGet)        (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                      LPUNKNOWN *) PURE;
   STDMETHOD (FX)                (THIS_ DWORD) PURE;

   // IVDctText2
   STDMETHOD (ITNApply)          (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ITNExpand)         (THIS_ DWORD, DWORD) PURE;
   STDMETHOD (ResultsGet2)       (THIS_ DWORD, DWORD, DWORD *, DWORD *,
                                     LPUNKNOWN *, DWORD*, DWORD*) PURE;
   STDMETHOD (ResultsSet)        (THIS_ DWORD, DWORD,
                                     LPUNKNOWN, DWORD, DWORD) PURE;
};
typedef IVDctText2A FAR * PIVDCTTEXT2A;


#ifdef _S_UNICODE
 #define IVDctText2      IVDctText2W
 #define IID_IVDctText2  IID_IVDctText2W
 #define PIVDCTTEXT2     PIVDCTTEXT2W

#else
 #define IVDctText2      IVDctText2A
 #define IID_IVDctText2  IID_IVDctText2A
 #define PIVDCTTEXT2     PIVDCTTEXT2A

#endif //_S_UNICODE



/*
 *  IVDctTextCache
 */
#undef   INTERFACE
#define  INTERFACE   IVDctTextCache

// {238004E3-F0C4-11d1-BED9-006008317CE8}
DEFINE_GUID(IID_IVDctTextCache,
0x238004e3, 0xf0c4, 0x11d1, 0xbe, 0xd9, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

DECLARE_INTERFACE_ (IVDctTextCache, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;
```

```
    // IVDctTextCache members
    STDMETHOD (CopyToBin)         (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (CopyToMemory)      (THIS_ DWORD, DWORD, PVOID*, DWORD*, LPUNKNOWN**, DWORD*) PURE;
    STDMETHOD (PasteFromBin)      (THIS_ DWORD, DWORD, DWORD) PURE;
    STDMETHOD (PasteFromMemory)   (THIS_ DWORD, DWORD, PVOID, DWORD, LPUNKNOWN*, DWORD) PURE;
};
typedef IVDctTextCache FAR * PIVDCTTEXTCACHE;


/*
 *  IVDctInvTextNorm
 */
#undef   INTERFACE
#define  INTERFACE   IVDctInvTextNormW

// {90A84EA0-6E51-11d0-9BC2-08005AFC3A41}
DEFINE_GUID(IID_IVDctInvTextNormW,
0x90a84ea0, 0x6e51, 0x11d0, 0x9b, 0xc2, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctInvTextNormW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctInvTextNorm members
    STDMETHOD     (OptionsEnum) (THIS_ PWSTR *, DWORD *) PURE;
    STDMETHOD     (OptionsGet)  (THIS_ PCWSTR, BOOL *) PURE;
    STDMETHOD     (OptionsSet)  (THIS_ PCWSTR, BOOL) PURE;
};

typedef IVDctInvTextNormW FAR * PIVDCTINVTEXTNORMW;


#undef   INTERFACE
#define  INTERFACE   IVDctInvTextNormA

// {90A84EA1-6E51-11d0-9BC2-08005AFC3A41}
DEFINE_GUID(IID_IVDctInvTextNormA,
0x90a84ea1, 0x6e51, 0x11d0, 0x9b, 0xc2, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctInvTextNormA, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctInvTextNorm members
    STDMETHOD     (OptionsEnum) (THIS_ PSTR *, DWORD *) PURE;
    STDMETHOD     (OptionsGet)  (THIS_ PCSTR, BOOL *) PURE;
    STDMETHOD     (OptionsSet)  (THIS_ PCSTR, BOOL) PURE;
};

typedef IVDctInvTextNormA FAR * PIVDCTINVTEXTNORMA;


#ifdef _S_UNICODE
 #define IVDctInvTextNorm     IVDctInvTextNormW
 #define IID_IVDctInvTextNorm IID_IVDctInvTextNormW
 #define PIVDCTINVTEXTNORM    PIVDCTINVTEXTNORMW

#else
 #define IVDctInvTextNorm     IVDctInvTextNormA
 #define IID_IVDctInvTextNorm IID_IVDctInvTextNormA
 #define PIVDCTINVTEXTNORM    PIVDCTINVTEXTNORMA

#endif // _S_UNICODE
```

```
/*
 *  IVDctAttributes
 */
#undef   INTERFACE
#define  INTERFACE   IVDctAttributesW

// {88AD7DC5-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctAttributesW,
0x88ad7dc5, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctAttributesW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctAttributes members
   STDMETHOD    (AutoGainEnableGet)   (THIS_ DWORD *) PURE;
   STDMETHOD    (AutoGainEnableSet)   (THIS_ DWORD) PURE;
   STDMETHOD    (AwakeStateGet)       (THIS_ DWORD *) PURE;
   STDMETHOD    (AwakeStateSet)       (THIS_ DWORD) PURE;
   STDMETHOD    (ThresholdGet)        (THIS_ DWORD *) PURE;
   STDMETHOD    (ThresholdSet)        (THIS_ DWORD) PURE;
   STDMETHOD    (EchoGet)             (THIS_ BOOL *) PURE;
   STDMETHOD    (EchoSet)             (THIS_ BOOL) PURE;
   STDMETHOD    (EnergyFloorGet)      (THIS_ WORD *) PURE;
   STDMETHOD    (EnergyFloorSet)      (THIS_ WORD) PURE;
   STDMETHOD    (RealTimeGet)         (THIS_ DWORD *) PURE;
   STDMETHOD    (RealTimeSet)         (THIS_ DWORD) PURE;
   STDMETHOD    (TimeOutGet)          (THIS_ DWORD *, DWORD *) PURE;
   STDMETHOD    (TimeOutSet)          (THIS_ DWORD, DWORD) PURE;
   STDMETHOD    (ModeGet)             (THIS_ DWORD *) PURE;
   STDMETHOD    (ModeSet)             (THIS_ DWORD) PURE;
   STDMETHOD    (MemoryGet)           (THIS_ VDCTMEMORY *) PURE;
   STDMETHOD    (MemorySet)           (THIS_ VDCTMEMORY *) PURE;
   STDMETHOD    (IsAnyoneDictating)   (THIS_ HWND, PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (MicrophoneGet)       (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (MicrophoneSet)       (THIS_ PCWSTR) PURE;
   STDMETHOD    (SpeakerGet)          (THIS_ PWSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (SpeakerSet)          (THIS_ PCWSTR) PURE;
};

typedef IVDctAttributesW FAR * PIVDCTATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   IVDctAttributesA

// {88AD7DC6-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctAttributesA,
0x88ad7dc6, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctAttributesA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctAttributes members
   STDMETHOD    (AutoGainEnableGet)   (THIS_ DWORD *) PURE;
   STDMETHOD    (AutoGainEnableSet)   (THIS_ DWORD) PURE;
   STDMETHOD    (AwakeStateGet)       (THIS_ DWORD *) PURE;
   STDMETHOD    (AwakeStateSet)       (THIS_ DWORD) PURE;
   STDMETHOD    (ThresholdGet)        (THIS_ DWORD *) PURE;
   STDMETHOD    (ThresholdSet)        (THIS_ DWORD) PURE;
   STDMETHOD    (EchoGet)             (THIS_ BOOL *) PURE;
   STDMETHOD    (EchoSet)             (THIS_ BOOL) PURE;
   STDMETHOD    (EnergyFloorGet)      (THIS_ WORD *) PURE;
   STDMETHOD    (EnergyFloorSet)      (THIS_ WORD) PURE;
   STDMETHOD    (RealTimeGet)         (THIS_ DWORD *) PURE;
   STDMETHOD    (RealTimeSet)         (THIS_ DWORD) PURE;
   STDMETHOD    (TimeOutGet)          (THIS_ DWORD *, DWORD *) PURE;
```

```
   STDMETHOD    (TimeOutSet)          (THIS_ DWORD, DWORD) PURE;
   STDMETHOD    (ModeGet)             (THIS_ DWORD *) PURE;
   STDMETHOD    (ModeSet)             (THIS_ DWORD) PURE;
   STDMETHOD    (MemoryGet)           (THIS_ VDCTMEMORY *) PURE;
   STDMETHOD    (MemorySet)           (THIS_ VDCTMEMORY *) PURE;
   STDMETHOD    (IsAnyoneDictating)   (THIS_ HWND, PSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (MicrophoneGet)       (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (MicrophoneSet)       (THIS_ PCSTR) PURE;
   STDMETHOD    (SpeakerGet)          (THIS_ PSTR, DWORD, DWORD *) PURE;
   STDMETHOD    (SpeakerSet)          (THIS_ PCSTR) PURE;
};

typedef IVDctAttributesA FAR * PIVDCTATTRIBUTESA;


#ifdef _S_UNICODE
 #define IVDctAttributes        IVDctAttributesW
 #define IID_IVDctAttributes    IID_IVDctAttributesW
 #define PIVDCTATTRIBUTES       PIVDCTATTRIBUTESW

#else
 #define IVDctAttributes        IVDctAttributesA
 #define IID_IVDctAttributes    IID_IVDctAttributesA
 #define PIVDCTATTRIBUTES       PIVDCTATTRIBUTESA

#endif // _S_UNICODE



/*
 * IVDctCommandsBuiltInW Interface
 */
// {D4E3F8E0-6521-11d1-8C35-006097DF5D01}
DEFINE_GUID(IID_IVDctCommandsBuiltInW,
0xd4e3f8e0, 0x6521, 0x11d1, 0x8c, 0x35, 0x0, 0x60, 0x97, 0xdf, 0x5d, 0x1);

// {8763AFD1-7ADE-11d1-BEA7-006008317CE8}
DEFINE_GUID(IID_IVDctCommandsBuiltInA,
0x8763afd1, 0x7ade, 0x11d1, 0xbe, 0xa7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

#undef   INTERFACE
#define  INTERFACE   IVDctCommandsBuiltInW

DECLARE_INTERFACE_ (IVDctCommandsBuiltInW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVDctCommandsBuiltInW members
   STDMETHOD (Activate) (THIS_ HWND, DWORD) PURE;
   STDMETHOD (Deactivate) (THIS) PURE;
   STDMETHOD (FromMemory) (THIS_ PVOID, DWORD) PURE;
   STDMETHOD (FromFile) (THIS_ PCWSTR) PURE;
   STDMETHOD (FromStream) (THIS_ IStream *) PURE;
   STDMETHOD (ListSet) (THIS_ PCWSTR, PVOID, DWORD) PURE;
   STDMETHOD (TextGet) (THIS_ LPWSTR*, DWORD*) PURE;
   STDMETHOD (ToMemory) (THIS_ PVOID*, DWORD*) PURE;
   STDMETHOD (ToFile) (THIS_ PCWSTR) PURE;
   STDMETHOD (ToStream) (THIS_ IStream *) PURE;
};

typedef IVDctCommandsBuiltInW * PIVDCTCOMMANDSBUILTINW;


#undef   INTERFACE
#define  INTERFACE   IVDctCommandsBuiltInA

DECLARE_INTERFACE_ (IVDctCommandsBuiltInA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
```

```
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVDctCommandsBuiltInA members
   STDMETHOD (Activate) (THIS_ HWND, DWORD) PURE;
   STDMETHOD (Deactivate) (THIS) PURE;
   STDMETHOD (FromMemory) (THIS_ PVOID, DWORD) PURE;
   STDMETHOD (FromFile) (THIS_ PCSTR) PURE;
   STDMETHOD (FromStream) (THIS_ IStream *) PURE;
   STDMETHOD (ListSet) (THIS_ PCSTR, PVOID, DWORD) PURE;
   STDMETHOD (TextGet) (THIS_ LPSTR*, DWORD*) PURE;
   STDMETHOD (ToMemory) (THIS_ PVOID*, DWORD*) PURE;
   STDMETHOD (ToFile) (THIS_ PCSTR) PURE;
   STDMETHOD (ToStream) (THIS_ IStream *) PURE;
};

typedef IVDctCommandsBuiltInA * PIVDCTCOMMANDSBUILTINA;

#ifdef _S_UNICODE
 #define IVDctCommandsBuiltIn        IVDctCommandsBuiltInW
 #define IID_IVDctCommandsBuiltIn    IID_IVDctCommandsBuiltInW
 #define PIVDCTCOMMANDSBUILTIN       PIVDCTCOMMANDSBUILTINW

#else
 #define IVDctCommandsBuiltIn        IVDctCommandsBuiltInA
 #define IID_IVDctCommandsBuiltIn    IID_IVDctCommandsBuiltInA
 #define PIVDCTCOMMANDSBUILTIN       PIVDCTCOMMANDSBUILTINA

#endif // _S_UNICODE



/*
 * IVDctCommandsApp Interface
 */
// {D4E3F8E1-6521-11d1-8C35-006097DF5D01}
DEFINE_GUID(IID_IVDctCommandsAppW,
0xd4e3f8e1, 0x6521, 0x11d1, 0x8c, 0x35, 0x0, 0x60, 0x97, 0xdf, 0x5d, 0x1);

// {8763AFD2-7ADE-11d1-BEA7-006008317CE8}
DEFINE_GUID(IID_IVDctCommandsAppA,
0x8763afd2, 0x7ade, 0x11d1, 0xbe, 0xa7, 0x0, 0x60, 0x8, 0x31, 0x7c, 0xe8);

typedef IVDctCommandsBuiltInA IVDctCommandsAppA;
typedef IVDctCommandsAppA * PIVDCTCOMMANDSAPPA;
typedef IVDctCommandsBuiltInW IVDctCommandsAppW;
typedef IVDctCommandsAppW * PIVDCTCOMMANDSAPPW;

#ifdef _S_UNICODE
 #define IVDctCommandsApp         IVDctCommandsAppW
 #define IID_IVDctCommandsApp     IID_IVDctCommandsAppW
 #define PIVDCTCOMMANDSAPP        PIVDCTCOMMANDSAPPW
#else
 #define IVDctCommandsApp         IVDctCommandsAppA
 #define IID_IVDctCommandsApp     IID_IVDctCommandsAppA
 #define PIVDCTCOMMANDSAPP        PIVDCTCOMMANDSAPPA
#endif // _S_UNICODE



/*
 *  IVDctCommands
 */
#undef   INTERFACE
#define  INTERFACE   IVDctCommandsW

// {A02C2CA0-AE50-11cf-833A-00AA00A21A29}
DEFINE_GUID(IID_IVDctCommandsW,
0xA02C2CA0, 0xAE50, 0x11cf, 0x83, 0x3A, 0x00, 0xAA, 0x00, 0xA2, 0x1A, 0x29);

DECLARE_INTERFACE_ (IVDctCommandsW, IUnknown) {
   // IUnknown members
```

```
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctCommands members
    STDMETHOD     (Add)       (THIS_ BOOL, DWORD, SDATA, DWORD *) PURE;
    STDMETHOD     (Remove)    (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD     (Get)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA*, DWORD *) PURE;
    STDMETHOD     (Set)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD     (EnableItem) (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD     (Num)       (THIS_ BOOL, DWORD *) PURE;
};

typedef IVDctCommandsW FAR * PIVDCTCOMMANDSW;


#undef  INTERFACE
#define INTERFACE   IVDctCommandsA

// {A02C2CA1-AE50-11cf-833A-00AA00A21A29}
DEFINE_GUID(IID_IVDctCommandsA,
0xA02C2CA1, 0xAE50, 0x11cf, 0x83, 0x3A, 0x00, 0xAA, 0x00, 0xA2, 0x1A, 0x29);

DECLARE_INTERFACE_ (IVDctCommandsA, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctCommands members
    STDMETHOD     (Add)       (THIS_ BOOL, DWORD, SDATA, DWORD *) PURE;
    STDMETHOD     (Remove)    (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD     (Get)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA*, DWORD *) PURE;
    STDMETHOD     (Set)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD     (EnableItem) (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD     (Num)       (THIS_ BOOL, DWORD *) PURE;
};

typedef IVDctCommandsA FAR * PIVDCTCOMMANDSA;


#ifdef _S_UNICODE
 #define IVDctCommands         IVDctCommandsW
 #define IID_IVDctCommands     IID_IVDctCommandsW
 #define PIVDCTCOMMANDS        PIVDCTCOMMANDSW

#else
 #define IVDctCommands         IVDctCommandsA
 #define IID_IVDctCommands     IID_IVDctCommandsA
 #define PIVDCTCOMMANDS        PIVDCTCOMMANDSA

#endif // _S_UNICODE



/*
 *  IVDctGlossary
 */
#undef  INTERFACE
#define INTERFACE   IVDctGlossaryW

// {A02C2CA2-AE50-11cf-833A-00AA00A21A29}
DEFINE_GUID(IID_IVDctGlossaryW,
0xA02C2CA2, 0xAE50, 0x11cf, 0x83, 0x3A, 0x00, 0xAA, 0x00, 0xA2, 0x1A, 0x29);

DECLARE_INTERFACE_ (IVDctGlossaryW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctGlossary members
```

```
    STDMETHOD    (Add)       (THIS_ BOOL, DWORD, SDATA, DWORD *) PURE;
    STDMETHOD    (Remove)    (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD    (Get)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA*, DWORD *) PURE;
    STDMETHOD    (Set)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD    (EnableItem) (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD    (Num)       (THIS_ BOOL, DWORD *) PURE;
};

typedef IVDctGlossaryW FAR * PIVDCTGLOSSARYW;


#undef   INTERFACE
#define  INTERFACE   IVDctGlossaryA

// {A02C2CA3-AE50-11cf-833A-00AA00A21A29}
DEFINE_GUID(IID_IVDctGlossaryA,
0xA02C2CA3, 0xAE50, 0x11cf, 0x83, 0x3A, 0x00, 0xAA, 0x00, 0xA2, 0x1A, 0x29);

DECLARE_INTERFACE_ (IVDctGlossaryA, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
    STDMETHOD_(ULONG,Release)   (THIS) PURE;

    // IVDctGlossary members
    STDMETHOD    (Add)       (THIS_ BOOL, DWORD, SDATA, DWORD *) PURE;
    STDMETHOD    (Remove)    (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD    (Get)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA*, DWORD *) PURE;
    STDMETHOD    (Set)       (THIS_ BOOL, DWORD, DWORD, DWORD, SDATA) PURE;
    STDMETHOD    (EnableItem) (THIS_ BOOL, DWORD, DWORD, DWORD) PURE;
    STDMETHOD    (Num)       (THIS_ BOOL, DWORD *) PURE;
};

typedef IVDctGlossaryA FAR * PIVDCTGLOSSARYA;


#ifdef _S_UNICODE
 #define IVDctGlossary        IVDctGlossaryW
 #define IID_IVDctGlossary    IID_IVDctGlossaryW
 #define PIVDCTGLOSSARY       PIVDCTGLOSSARYW

#else
 #define IVDctGlossary        IVDctGlossaryA
 #define IID_IVDctGlossary    IID_IVDctGlossaryA
 #define PIVDCTGLOSSARY       PIVDCTGLOSSARYA

#endif // _S_UNICODE



/*
 *  IVDctDialog
 */
#undef   INTERFACE
#define  INTERFACE   IVDctDialogsW

// {88AD7DC7-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctDialogsW,
0x88ad7dc7, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctDialogsW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
    STDMETHOD_(ULONG,Release)     (THIS) PURE;

    // IVDctDialogs members
    STDMETHOD (AboutDlg)          (THIS_ HWND, PCWSTR) PURE;
    STDMETHOD (GeneralDlg)        (THIS_ HWND, PCWSTR) PURE;
    STDMETHOD (LexiconDlg)        (THIS_ HWND, PCWSTR) PURE;
    STDMETHOD (TrainGeneralDlg)   (THIS_ HWND, PCWSTR) PURE;
```

```
   STDMETHOD (TrainMicDlg)       (THIS_ HWND, PCWSTR) PURE;
};

typedef IVDctDialogsW FAR * PIVDCTDIALOGSW;


#undef   INTERFACE
#define  INTERFACE   IVDctDialogsA

// {88AD7DC8-67D5-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(IID_IVDctDialogsA,
0x88ad7dc8, 0x67d5, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);

DECLARE_INTERFACE_ (IVDctDialogsA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // IVDctDialogs members
   STDMETHOD (AboutDlg)          (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)        (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)        (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainGeneralDlg)   (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TrainMicDlg)       (THIS_ HWND, PCSTR) PURE;
};

typedef IVDctDialogsA FAR * PIVDCTDIALOGSA;


#ifdef _S_UNICODE
 #define IVDctDialogs       IVDctDialogsW
 #define IID_IVDctDialogs   IID_IVDctDialogsW
 #define PIVDCTDIALOGS      PIVDCTDIALOGSW

#else
 #define IVDctDialogs       IVDctDialogsA
 #define IID_IVDctDialogs   IID_IVDctDialogsA
 #define PIVDCTDIALOGS      PIVDCTDIALOGSA

#endif // _S_UNICODE


#undef   INTERFACE
#define  INTERFACE   IVDctGUI

// {8953F1A0-7E80-11cf-8D15-00A0C9034A7E}
DEFINE_GUID(IID_IVDctGUI,
0x8953f1a0, 0x7e80, 0x11cf, 0x8d, 0x15, 0x0, 0xa0, 0xc9, 0x3, 0x4a, 0x7e);

DECLARE_INTERFACE_ (IVDctGUI, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)     (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)      (THIS) PURE;
   STDMETHOD_(ULONG,Release)     (THIS) PURE;

   // IVDctDialogs members
   STDMETHOD (SetSelRect)        (THIS_ RECT *) PURE;
   STDMETHOD (FlagsSet)          (THIS_ DWORD) PURE;
   STDMETHOD (FlagsGet)          (THIS_ DWORD *) PURE;
};

typedef IVDctGUI FAR * PIVDCTGUI;


/************************************************************************
class guids */

// {25522CA0-67CE-11cf-9B8B-08005AFC3A41}
DEFINE_GUID(CLSID_VDct,
0x35522ca0, 0x67ce, 0x11cf, 0x9b, 0x8b, 0x8, 0x0, 0x5a, 0xfc, 0x3a, 0x41);
```

```
/*****************************************************************************
High-Level text-to-speech API
*/


/*****************************************************************************
defines */

#define  ONE                    (1)

// dwFlags parameter of IVoiceText::Register
#define  VTXTF_ALLMESSAGES      (ONE<<0)

/*
 *   dwFlags parameter of IVoiceText::Speak
 */

// type of speech
#define  VTXTST_STATEMENT       0x00000001
#define  VTXTST_QUESTION        0x00000002
#define  VTXTST_COMMAND         0x00000004
#define  VTXTST_WARNING         0x00000008
#define  VTXTST_READING         0x00000010
#define  VTXTST_NUMBERS         0x00000020
#define  VTXTST_SPREADSHEET     0x00000040

// priorities
#define  VTXTSP_VERYHIGH        0x00000080
#define  VTXTSP_HIGH            0x00000100
#define  VTXTSP_NORMAL          0x00000200

/*****************************************************************************
typedefs */

// possible parameter to IVoiceText::Register
typedef struct {
    DWORD   dwDevice;
    DWORD   dwEnable;
    DWORD   dwSpeed;
    GUID    gModeID;
} VTSITEINFO, *PVTSITEINFO;


/*****************************************************************************
interfaces */

/*
 *  IVCmdNotifySink
 */
#undef   INTERFACE
#define  INTERFACE   IVTxtNotifySinkW

// {FD3A2430-E090-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtNotifySinkW, 0xfd3a2430, 0xe090, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtNotifySinkW, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVTxtNotifySinkW members
   STDMETHOD (AttribChanged)     (THIS_ DWORD) PURE;
   STDMETHOD (Visual)            (THIS_ WCHAR, WCHAR, DWORD, PTTSMOUTH) PURE;
   STDMETHOD (Speak)             (THIS_ PWSTR, PWSTR, DWORD) PURE;
   STDMETHOD (SpeakingStarted)   (THIS) PURE;
   STDMETHOD (SpeakingDone)      (THIS) PURE;
```

```
};

typedef IVTxtNotifySinkW FAR * PIVTXTNOTIFYSINKW;


#undef   INTERFACE
#define  INTERFACE   IVTxtNotifySinkA

// {D2C840E0-E092-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtNotifySinkA, 0xd2c840e0, 0xe092, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtNotifySinkA, IUnknown) {

   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVTxtNotifySinkA members
   STDMETHOD (AttribChanged)     (THIS_ DWORD) PURE;
   STDMETHOD (Visual)            (THIS_ WCHAR, CHAR, DWORD, PTTSMOUTH) PURE;
   STDMETHOD (Speak)             (THIS_ PSTR, PSTR, DWORD) PURE;
   STDMETHOD (SpeakingStarted)   (THIS) PURE;
   STDMETHOD (SpeakingDone)      (THIS) PURE;
};

typedef IVTxtNotifySinkA FAR * PIVTXTNOTIFYSINKA;


#ifdef _S_UNICODE
 #define IVTxtNotifySink        IVTxtNotifySinkW
 #define IID_IVTxtNotifySink    IID_IVTxtNotifySinkW
 #define PIVTXTNOTIFYSINK       PIVTXTNOTIFYSINKW

#else
 #define IVTxtNotifySink        IVTxtNotifySinkA
 #define IID_IVTxtNotifySink    IID_IVTxtNotifySinkA
 #define PIVTXTNOTIFYSINK       PIVTXTNOTIFYSINKA

#endif // _S_UNICODE



/*
 *  IVoiceText
 */
#undef   INTERFACE
#define  INTERFACE   IVoiceTextW

// {C4FE8740-E093-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVoiceTextW, 0xc4fe8740, 0xe093, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);

DECLARE_INTERFACE_ (IVoiceTextW, IUnknown) {
    // IUnknown members
    STDMETHOD(QueryInterface)    (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef)     (THIS) PURE;
    STDMETHOD_(ULONG,Release)    (THIS) PURE;

    // IVoiceText members

    STDMETHOD (Register)          (THIS_ PCWSTR, PCWSTR,
                   PIVTXTNOTIFYSINK, GUID,
                   DWORD, PVTSITEINFO) PURE;
    STDMETHOD (Speak)             (THIS_ PCWSTR, DWORD, PCWSTR) PURE;
    STDMETHOD (StopSpeaking)      (THIS) PURE;
    STDMETHOD (AudioFastForward)  (THIS) PURE;
    STDMETHOD (AudioPause)        (THIS) PURE;
    STDMETHOD (AudioResume)       (THIS) PURE;
    STDMETHOD (AudioRewind)       (THIS) PURE;
};
```

```
typedef IVoiceTextW FAR * PIVOICETEXTW;


#undef   INTERFACE
#define  INTERFACE   IVoiceTextA

// {E1B7A180-E093-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVoiceTextA, 0xe1b7a180, 0xe093, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);

DECLARE_INTERFACE_ (IVoiceTextA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVoiceText members

    STDMETHOD (Register)         (THIS_ PCSTR, PCSTR,
                  PIVTXTNOTIFYSINK, GUID,
                  DWORD, PVTSITEINFO) PURE;
    STDMETHOD (Speak)            (THIS_ PCSTR, DWORD, PCSTR) PURE;
    STDMETHOD (StopSpeaking)     (THIS) PURE;
    STDMETHOD (AudioFastForward) (THIS) PURE;
    STDMETHOD (AudioPause)       (THIS) PURE;
    STDMETHOD (AudioResume)      (THIS) PURE;
    STDMETHOD (AudioRewind)      (THIS) PURE;
};

typedef IVoiceTextA FAR * PIVOICETEXTA;


#ifdef _S_UNICODE
 #define IVoiceText      IVoiceTextW
 #define IID_IVoiceText  IID_IVoiceTextW
 #define PIVOICETEXT     PIVOICETEXTW

#else
 #define IVoiceText      IVoiceTextA
 #define IID_IVoiceText  IID_IVoiceTextA
 #define PIVOICETEXT     PIVOICETEXTA

#endif //_S_UNICODE



/*
 *  IVTxtAttributes
 */
#undef   INTERFACE
#define  INTERFACE   IVTxtAttributesW

// {6A8D6140-E095-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtAttributesW, 0x6a8d6140, 0xe095, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtAttributesW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVTxtAttributes members
   STDMETHOD (DeviceGet)          (THIS_ DWORD *) PURE;
   STDMETHOD (DeviceSet)          (THIS_ DWORD) PURE;
   STDMETHOD (EnabledGet)         (THIS_ DWORD *) PURE;
   STDMETHOD (EnabledSet)         (THIS_ DWORD) PURE;
   STDMETHOD (IsSpeaking)         (THIS_ BOOL *) PURE;
   STDMETHOD (SpeedGet)           (THIS_ DWORD *) PURE;
   STDMETHOD (SpeedSet)           (THIS_ DWORD) PURE;
   STDMETHOD (TTSModeGet)         (THIS_ GUID *) PURE;
   STDMETHOD (TTSModeSet)         (THIS_ GUID) PURE;
};
```

```
typedef IVTxtAttributesW FAR * PIVTXTATTRIBUTESW;


#undef   INTERFACE
#define  INTERFACE   IVTxtAttributesA

// {8BE9CC30-E095-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtAttributesA, 0x8be9cc30, 0xe095, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtAttributesA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVTxtAttributes members
   STDMETHOD (DeviceGet)          (THIS_ DWORD *) PURE;
   STDMETHOD (DeviceSet)          (THIS_ DWORD) PURE;
   STDMETHOD (EnabledGet)         (THIS_ DWORD *) PURE;
   STDMETHOD (EnabledSet)         (THIS_ DWORD) PURE;
   STDMETHOD (IsSpeaking)         (THIS_ BOOL *) PURE;
   STDMETHOD (SpeedGet)           (THIS_ DWORD *) PURE;
   STDMETHOD (SpeedSet)           (THIS_ DWORD) PURE;
   STDMETHOD (TTSModeGet)         (THIS_ GUID *) PURE;
   STDMETHOD (TTSModeSet)         (THIS_ GUID) PURE;
};

typedef IVTxtAttributesA FAR * PIVTXTATTRIBUTESA;


#ifdef _S_UNICODE
 #define IVTxtAttributes        IVTxtAttributesW
 #define IID_IVTxtAttributes    IID_IVTxtAttributesW
 #define PIVTXTATTRIBUTES       PIVTXTATTRIBUTESW

#else
 #define IVTxtAttributes        IVTxtAttributesA
 #define IID_IVTxtAttributes    IID_IVTxtAttributesA
 #define PIVTXTATTRIBUTES       PIVTXTATTRIBUTESA

#endif // _S_UNICODE



/*
 *  IVTxtDialog
 */
#undef   INTERFACE
#define  INTERFACE   IVTxtDialogsW

// {D6469210-E095-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtDialogsW, 0xd6469210, 0xe095, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtDialogsW, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IVTxtDialogs members

   STDMETHOD (AboutDlg)        (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (LexiconDlg)      (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (GeneralDlg)      (THIS_ HWND, PCWSTR) PURE;
   STDMETHOD (TranslateDlg)    (THIS_ HWND, PCWSTR) PURE;
};

typedef IVTxtDialogsW FAR * PIVTXTDIALOGSW;
```

```
#undef   INTERFACE
#define  INTERFACE   IVTxtDialogsA

// {E8F6FA20-E095-11cd-A166-00AA004CD65C}
DEFINE_GUID(IID_IVTxtDialogsA, 0xe8f6fa20, 0xe095, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6,
0x5c);

DECLARE_INTERFACE_ (IVTxtDialogsA, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)  (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   // IVTxtDialogs members
   STDMETHOD (AboutDlg)       (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (LexiconDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (GeneralDlg)     (THIS_ HWND, PCSTR) PURE;
   STDMETHOD (TranslateDlg)   (THIS_ HWND, PCSTR) PURE;
};

typedef IVTxtDialogsA FAR * PIVTXTDIALOGSA;


#ifdef _S_UNICODE
 #define IVTxtDialogs       IVTxtDialogsW
 #define IID_IVTxtDialogs   IID_IVTxtDialogsW
 #define PIVTXTDIALOGS      PIVTXTDIALOGSW

#else
 #define IVTxtDialogs       IVTxtDialogsA
 #define IID_IVTxtDialogs   IID_IVTxtDialogsA
 #define PIVTXTDIALOGS      PIVTXTDIALOGSA

#endif // _S_UNICODE



/************************************************************************
class guids */

// {080EB9D0-E096-11cd-A166-00AA004CD65C}
// DEFINE_GUID(CLSID_VTxt, 0x80eb9d0, 0xe096, 0x11cd, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);
// {F1DC95A0-0BA7-11ce-A166-00AA004CD65C}
DEFINE_GUID(CLSID_VTxt,
0xf1dc95a0, 0xba7, 0x11ce, 0xa1, 0x66, 0x0, 0xaa, 0x0, 0x4c, 0xd6, 0x5c);


// {70618F72-D1ED-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_STGramComp,
0x70618f72, 0xd1ed, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

// {EFD0E6BA-DB5F-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_STTTSQueue,
0xefd0e6ba, 0xdb5f, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

// {B3613D9F-E26E-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_STLog,
0xb3613d9f, 0xe26e, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

// {B3613DA0-E26E-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_AudioSourceLog,
0xb3613da0, 0xe26e, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

// {2EC5A8A5-E65B-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_AudioSourceTel,
0x2ec5a8a5, 0xe65b, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

// {2EC5A8A6-E65B-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(CLSID_AudioDestTel,
0x2ec5a8a6, 0xe65b, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);
```

```
// {61935832-FC85-11d0-8FAE-08002BE4E62A}
DEFINE_GUID(CLSID_STRecord,
0x61935832, 0xfc85, 0x11d0, 0x8f, 0xae, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

/* interfaces */
// ISTRecord Interface

#undef   INTERFACE
#define  INTERFACE   ISTRecord

// {61935833-FC85-11d0-8FAE-08002BE4E62A}
DEFINE_GUID(IID_ISTRecord,
0x61935833, 0xfc85, 0x11d0, 0x8f, 0xae, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

typedef struct {
   DWORD       dwBetweenUtt;  // acceptable time between utterance, milliseconds
   DWORD       dwInitialBuf;  // milliseconds of audio in the initial buffer
   DWORD       dwReallocBuf;  // amount to increase a buffer every time new audio
   DWORD       dwMaxBuf;      // maximum number of milliseconds that buffer can be
   DWORD       dwNoAnswer;    // milliseconds of silence before get no-answer
} STRECORDPARAM, *PSTRECORDPARAM;

DECLARE_INTERFACE_ (ISTRecord, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTGramComp members
   STDMETHOD (Init) (THIS_ LPUNKNOWN, DWORD, LPUNKNOWN) PURE;
   STDMETHOD (ParamsSet) (THIS_ PSTRECORDPARAM) PURE;
   STDMETHOD (Start) (THIS) PURE;
   STDMETHOD (Stop) (THIS_ DWORD) PURE;
   STDMETHOD (GetWAV) (THIS_ PVOID*, DWORD*) PURE;
   STDMETHOD (ClearWAV) (THIS) PURE;
   };

typedef ISTRecord FAR * PISTRECORD;


// ISTRecordNotifySink Interface

#undef   INTERFACE
#define  INTERFACE   ISTRecordNotifySink

// {61935834-FC85-11d0-8FAE-08002BE4E62A}
DEFINE_GUID(IID_ISTRecordNotifySink,
0x61935834, 0xfc85, 0x11d0, 0x8f, 0xae, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (ISTRecordNotifySink, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTGramComp members
   STDMETHOD (AutoStop) (THIS) PURE;
   STDMETHOD (VU) (THIS_ WORD) PURE;
   };

typedef ISTRecordNotifySink FAR * PISTRECORDNOTIFYSINK;


// ISTGramComp Interface

#undef   INTERFACE
#define  INTERFACE   ISTGramComp

// {70618F73-D1ED-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_ISTGramComp,
0x70618f73, 0xdled, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);
```

```
DECLARE_INTERFACE_ (ISTGramComp, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTGramComp members
   STDMETHOD (FromMemory) (THIS_ PVOID, DWORD) PURE;
   STDMETHOD (FromResource) (THIS_ HINSTANCE, DWORD) PURE;
   STDMETHOD (FromFile) (THIS_ PCWSTR) PURE;
   STDMETHOD (FromStream) (THIS_ IStream *) PURE;
   STDMETHOD (Compile) (THIS_ LPWSTR*, DWORD*) PURE;
   STDMETHOD (IsCompiled) (THIS_ BOOL*) PURE;
   STDMETHOD (ToMemory) (THIS_ PVOID*, DWORD*) PURE;
   STDMETHOD (ToFile) (THIS_ PCWSTR) PURE;
   STDMETHOD (ToStream) (THIS_ IStream *) PURE;
   STDMETHOD (TextGet) (THIS_ LPWSTR*, DWORD*) PURE;
   STDMETHOD (LanguageGet) (THIS_ LANGUAGEW*) PURE;
   STDMETHOD (GrammarFormatGet) (THIS_ SRGRMFMT*) PURE;
   STDMETHOD (GrammarDataGet) (THIS_ BOOL, PVOID*, DWORD*) PURE;
   STDMETHOD (GrammarDataSet) (THIS_ PVOID, DWORD) PURE;
   STDMETHOD (AutoList) (THIS_ PISRGRAMCFGW) PURE;
   STDMETHOD (PhraseParse) (THIS_ PSRPHRASEW, DWORD *, LPWSTR *, DWORD *) PURE;
   STDMETHOD (GrammarLoad) (THIS_ LPUNKNOWN, PVOID, IID, LPUNKNOWN *) PURE;
   STDMETHOD (PhraseParseAlt) (THIS_ DWORD, DWORD *, LPWSTR *, DWORD *) PURE;
   STDMETHOD (PhraseParseRule) (THIS_ PCWSTR, PSRPHRASEW, DWORD *, LPWSTR *, DWORD *) PURE;
   };

typedef ISTGramComp FAR * PISTGRAMCOMP;


// ISTTTSQueue Interface

#undef   INTERFACE
#define  INTERFACE   ISTTTSQueue

// {EFD0E6BB-DB5F-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_ISTTTSQueue,
0xefd0e6bb, 0xdb5f, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (ISTTTSQueue, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTTTSQueue members
   STDMETHOD (WaveAddFromFile) (THIS_ PCWSTR, PCWSTR) PURE;
   STDMETHOD (WaveAddFromStream) (THIS_ PCWSTR, IStream *) PURE;
   STDMETHOD (WaveAddFromMemory) (THIS_ PCWSTR, PVOID, DWORD) PURE;
   STDMETHOD (WaveAddFromResource) (THIS_ PCWSTR, HMODULE, DWORD) PURE;
   STDMETHOD (WaveAddFromList) (THIS_ PCWSTR, HMODULE) PURE;
   STDMETHOD (WaveRemove) (THIS_ PCWSTR) PURE;
   STDMETHOD (WaveAudioDestSet) (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (WaveLevelSet) (THIS_ DWORD) PURE;
   STDMETHOD (WaveLevelGet) (THIS_ DWORD *) PURE;
   STDMETHOD (TTSAdd) (THIS_ PCWSTR, LPUNKNOWN) PURE;
   STDMETHOD (TTSRemove) (THIS_ PCWSTR) PURE;
   STDMETHOD (TTSGet) (THIS_ PCWSTR, LPUNKNOWN *) PURE;
   STDMETHOD (Speak) (THIS_ PCWSTR, PCWSTR, DWORD) PURE;
   STDMETHOD (TextData) (THIS_ VOICECHARSET, DWORD, SDATA, PCWSTR, DWORD) PURE;
   STDMETHOD (AudioPause) (THIS) PURE;
   STDMETHOD (AudioResume) (THIS) PURE;
   STDMETHOD (AudioReset) (THIS) PURE;
   STDMETHOD (Register) (THIS_ PVOID, REFIID) PURE;
   STDMETHOD (UseLogging) (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (IsSpeaking) (THIS_ DWORD*) PURE;
   };

typedef ISTTTSQueue FAR * PISTTTSQUEUE;
```

```
// ISTTTSQueueNotifySink Interface

#undef   INTERFACE
#define  INTERFACE   ISTTTSQueueNotifySink

// {8412FA0C-DB6A-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_ISTTTSQueueNotifySink,
0x8412fa0c, 0xdb6a, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (ISTTTSQueueNotifySink, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTTTSQueueNotifySink members
   STDMETHOD (Start) (THIS) PURE;
   STDMETHOD (Stop) (THIS) PURE;
   STDMETHOD (SpeakID) (THIS_ DWORD) PURE;
   STDMETHOD (Error) (THIS_ HRESULT) PURE;
   STDMETHOD (Bookmark) (THIS_ DWORD) PURE;
   };

typedef ISTTTSQueueNotifySink FAR * PISTTTSQUEUENOTIFYSINK;


// ISTLog Interface

#undef   INTERFACE
#define  INTERFACE   ISTLog

// {B3613DA1-E26E-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_ISTLog,
0xb3613da1, 0xe26e, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (ISTLog, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // ISTLog members
   STDMETHOD (ToFile) (THIS_ PCWSTR) PURE;
   STDMETHOD (ToStream) (THIS_ IStream *) PURE;
   STDMETHOD (Text) (THIS_ PCWSTR, PCWSTR, WORD) PURE;
   STDMETHOD (Data) (THIS_ PCWSTR, PCWSTR, WORD, PVOID, DWORD) PURE;
   STDMETHOD (MaxDetailSet) (THIS_ WORD) PURE;
   STDMETHOD (MaxDetailGet) (THIS_ WORD*) PURE;
   STDMETHOD (ShowOutput) (THIS_ BOOL) PURE;
   };

typedef ISTLog FAR * PISTLOG;


// IAudioSourceLog Interface

#undef   INTERFACE
#define  INTERFACE   IAudioSourceLog

// {B3613DA2-E26E-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_IAudioSourceLog,
0xb3613da2, 0xe26e, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (IAudioSourceLog, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IAudioSourceLog members
```

```
   STDMETHOD (AudioSource) (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (UseLogging) (THIS_ LPUNKNOWN) PURE;
   };

typedef IAudioSourceLog FAR * PIAUDIOSOURCELOG;


// IAudioTel Interface

#undef   INTERFACE
#define  INTERFACE   IAudioTel

// {2EC5A8A7-E65B-11d0-8FAC-08002BE4E62A}
DEFINE_GUID(IID_IAudioTel,
0x2ec5a8a7, 0xe65b, 0x11d0, 0x8f, 0xac, 0x8, 0x0, 0x2b, 0xe4, 0xe6, 0x2a);

DECLARE_INTERFACE_ (IAudioTel, IUnknown) {
   // IUnknown members
   STDMETHOD(QueryInterface)   (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)    (THIS) PURE;
   STDMETHOD_(ULONG,Release)   (THIS) PURE;

   // IAudioTel members
   STDMETHOD (AudioObject) (THIS_ LPUNKNOWN) PURE;
   STDMETHOD (WaveFormatSet) (THIS_ SDATA) PURE;
   };

typedef IAudioTel FAR * PIAUDIOTEL;



/**************************************************************************
External speech synthesizer - This is for using external, hardware
based speech synthesizers with SAPI */


/**************************************************************************
defines */


// Synthesizer Attributes
#define  TTS_LANGUAGE          SETBIT(0)
#define  TTS_VOICE             SETBIT(1)
#define  TTS_GENDER            SETBIT(2)
#define  TTS_VOLUME            SETBIT(3)
#define  TTS_PITCH             SETBIT(4)
#define  TTS_SPEED             SETBIT(5)
#define  TTS_ABBREVIATION      SETBIT(6)
#define  TTS_PUNCTUATION       SETBIT(7)
#define  TTS_PAUSEWORD         SETBIT(8)
#define  TTS_PAUSEPHRASE       SETBIT(9)
#define  TTS_PAUSESENTENCE     SETBIT(10)
#define  TTS_SPELLING          SETBIT(11)
#define  TTS_QUALITY           SETBIT(12)
#define  TTS_FRICATION         SETBIT(13)
#define  TTS_ASPIRATION        SETBIT(14)
#define  TTS_INTONATION        SETBIT(15)


// Attribute minimums and maximums
#define  TTSATTR_MINPAUSEWORD       0x00000000
#define  TTSATTR_MAXPAUSEWORD       0xffffffff
#define  TTSATTR_MINPAUSEPHRASE     0x00000000
#define  TTSATTR_MAXPAUSEPHRASE     0xffffffff
#define  TTSATTR_MINPAUSESENTENCE   0x00000000
#define  TTSATTR_MAXPAUSESENTENCE   0xffffffff
#define  TTSATTR_MINASPIRATION      0x00000000
#define  TTSATTR_MAXASPIRATION      0xffffffff
#define  TTSATTR_MINFRICATION       0x00000000
```

```
#define  TTSATTR_MAXFRICATION      0xffffffff
#define  TTSATTR_MININTONATION     0x00000000
#define  TTSATTR_MAXINTONATION     0xffffffff


// ITTSNotifySink
#define TTSNSAC_LANGUAGE         100
#define TTSNSAC_VOICE            101
#define TTSNSAC_GENDER           102
#define TTSNSAC_ABBREVIATION     103
#define TTSNSAC_PUNCTUATION      104
#define TTSNSAC_PAUSEWORD        105
#define TTSNSAC_PAUSEPHRASE      106
#define TTSNSAC_PAUSESENTENCE    107
#define TTSNSAC_SPELLING         108
#define TTSNSAC_QUALITY          109
#define TTSNSAC_FRICATION        110
#define TTSNSAC_ASPIRATION       111
#define TTSNSAC_INTONATION       112


// Interfaces
#define TTSI_ITTSEXTERNALSYNTHESIZER           SETBIT(4)


// TTSERR Result codes
#define TTSERR_SYNTHESIZERBUSY                 TTSERROR(100)
#define TTSERR_ALREADYDISPLAYED                TTSERROR(101)
#define TTSERR_INVALIDATTRIB                   TTSERROR(102)
#define TTSERR_SYNTHESIZERACCESSERROR          TTSERROR(103)
#define TTSERR_DRIVERERROR                     TTSERROR(104)
#define TTSERR_UNRECOVERABLEERROR              TTSERROR(105)
#define TTSERR_DRIVERACCESSERROR               TTSERROR(106)
#define TTSERR_BUFFERTOOSMALL                  TTSERROR(107)
#define TTSERR_DRIVERNOTFOUND                  TTSERROR(108)
#define TTSERR_CANNOTREGISTER                  TTSERROR(109)
#define TTSERR_LANGUAGENOTSUPPORTED            TTSERROR(110)


/*****************************************************************************
Class IDs */

// {479DFDCB-01A8-11d1-A7A5-00C04FBBE8AB}
DEFINE_GUID(CLSID_SynthAudioDest,
0x479dfdcb, 0x1a8, 0x11d1, 0xa7, 0xa5, 0x0, 0xc0, 0x4f, 0xbb, 0xe8, 0xab);

// {26C328C1-F26F-11d0-9DFF-006097622195}
DEFINE_GUID(CLSID_SynthEngine,
0x26c328c1, 0xf26f, 0x11d0, 0x9d, 0xff, 0x0, 0x60, 0x97, 0x62, 0x21, 0x95);

// {AC8FCCA1-FFD7-11d0-9E22-006097622195}
DEFINE_GUID(CLSID_SynthEngineEnumerator,
0xac8fcca1, 0xffd7, 0x11d0, 0x9e, 0x22, 0x0, 0x60, 0x97, 0x62, 0x21, 0x95);


/*****************************************************************************
typedefs */

// Definition of Preset
typedef struct
{
   LANGID          Language;
   DWORD           Voice;
   DWORD           Gender;

   DWORD           Volume;
   DWORD           Pitch;
   DWORD           Speed;

   DWORD           Abbreviation;
   DWORD           Punctuation;
```

```
   DWORD           PauseWord;
   DWORD           PausePhrase;
   DWORD           PauseSentence;

   DWORD           Spelling;
   DWORD           Quality;

   DWORD           Frication;
   DWORD           Aspiration;
   DWORD           Intonation;
}
TTSPRESET, *PTTSPRESET;


/****************************************************************************
interfaces */


/*
 * ITTSExternalSynthesizer
 */


#undef   INTERFACE
#define  INTERFACE   ITTSExternalSynthesizerW

// {A3F8B9C0-2010-11d1-A306-006097622195}
DEFINE_GUID(IID_ITTSExternalSynthesizerW,
0xa3f8b9c0, 0x2010, 0x11d1, 0xa3, 0x6, 0x0, 0x60, 0x97, 0x62, 0x21, 0x95);

DECLARE_INTERFACE_(ITTSExternalSynthesizerW, IUnknown)
{
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID  * ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;

   STDMETHOD (AttributeGet) (THIS_ DWORD dwAttribute, DWORD *pdwValue)   PURE;
   STDMETHOD (AttributeSet) (THIS_ DWORD dwAttribute, DWORD dwValue)     PURE;
   STDMETHOD (PresetGet)    (THIS_ PTTSPRESET pPreset, DWORD *pdwResult) PURE;
   STDMETHOD (PresetSet)    (THIS_ TTSPRESET Preset, DWORD *pdwResult)   PURE;

   STDMETHOD (GetAttributeElements) (THIS_ DWORD dwAttr, DWORD *pdwNumElems) PURE;
   STDMETHOD (GetAttributeRange)    (THIS_ DWORD dwAttr, DWORD *pdwMin, DWORD *pdwMax) PURE;
   STDMETHOD (GetAttributeValue)    (THIS_ DWORD dwAttr, DWORD dwIndex, DWORD *pdwValue, PWSTR szText,
DWORD dwMaxChars) PURE;
   STDMETHOD (GetAttributeIndex)    (THIS_ DWORD dwAttr, DWORD dwValue, DWORD *pdwIndex, PWSTR szText,
DWORD dwMaxChars) PURE;

   STDMETHOD (AudioFlush) (THIS) PURE;

   STDMETHOD (TextDataInterrupt)(THIS_ VOICECHARSET eCharacterSet, DWORD dwFlags, SDATA dText, PVOID
pNotifyInterface, IID IIDNotifyInterface) PURE;
   STDMETHOD (TextDataCritical) (THIS_ SDATA) PURE;

   STDMETHOD (MiscellaneousDlg) (THIS_ HWND hWndParent, PCWSTR pszTitle) PURE;
};
typedef ITTSExternalSynthesizerW  *PITTSEXTERNALSYNTHESIZERW;


#undef   INTERFACE
#define  INTERFACE   ITTSExternalSynthesizerA

// {3D11C963-F210-11d0-A788-00C04FBBE8AB}
DEFINE_GUID(IID_ITTSExternalSynthesizerA,
0x3d11c963, 0xf210, 0x11d0, 0xa7, 0x88, 0x0, 0xc0, 0x4f, 0xbb, 0xe8, 0xab);

DECLARE_INTERFACE_(ITTSExternalSynthesizerA, IUnknown)
{
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID  * ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;
```

```
   STDMETHOD (AttributeGet) (THIS_ DWORD dwAttribute, DWORD *pdwValue)    PURE;
   STDMETHOD (AttributeSet) (THIS_ DWORD dwAttribute, DWORD dwValue)      PURE;
   STDMETHOD (PresetGet)    (THIS_ PTTSPRESET pPreset, DWORD *pdwResult) PURE;
   STDMETHOD (PresetSet)    (THIS_ TTSPRESET Preset, DWORD *pdwResult)   PURE;

   STDMETHOD (GetAttributeElements) (THIS_ DWORD dwAttr, DWORD *pdwNumElems) PURE;
   STDMETHOD (GetAttributeRange)    (THIS_ DWORD dwAttr, DWORD *pdwMin, DWORD *pdwMax) PURE;
   STDMETHOD (GetAttributeValue)    (THIS_ DWORD dwAttr, DWORD dwIndex, DWORD *pdwValue, PSTR szText,
DWORD dwMaxChars) PURE;
   STDMETHOD (GetAttributeIndex)    (THIS_ DWORD dwAttr, DWORD dwValue, DWORD *pdwIndex, PSTR szText,
DWORD dwMaxChars) PURE;

   STDMETHOD (AudioFlush) (THIS) PURE;

   STDMETHOD (TextDataInterrupt)(THIS_ VOICECHARSET eCharacterSet, DWORD dwFlags, SDATA dText, PVOID
pNotifyInterface, IID IIDNotifyInterface) PURE;
   STDMETHOD (TextDataCritical) (THIS_ SDATA) PURE;

   STDMETHOD (MiscellaneousDlg) (THIS_ HWND hWndParent, PCSTR pszTitle) PURE;
};
typedef ITTSExternalSynthesizerA  *PITTSEXTERNALSYNTHESIZERA;


#ifdef _S_UNICODE
 #define ITTSExternalSynthesizer            ITTSExternalSynthesizerW
 #define IID_ITTSExternalSynthesizer        IID_ITTSExternalSynthesizerW
 #define PITTSEXTERNALSYNTHESIZER           PITTSEXTERNALSYNTHESIZERW

#else
 #define ITTSExternalSynthesizer           ITTSExternalSynthesizerA
 #define IID_ITTSExternalSynthesizer       IID_ITTSExternalSynthesizerA
 #define PITTSEXTERNALSYNTHESIZER          PITTSEXTERNALSYNTHESIZERA

#endif   // _S_UNICODE



/*
 * IAudioExternalSynthesizer
 */

#undef   INTERFACE
#define  INTERFACE   IAudioExternalSynthesizer

// {33E0E76B-70CE-11d1-A826-00C04FBBE8AB}
DEFINE_GUID(IID_IAudioExternalSynthesizer,
0x33e0e76b, 0x70ce, 0x11d1, 0xa8, 0x26, 0x0, 0xc0, 0x4f, 0xbb, 0xe8, 0xab);

DECLARE_INTERFACE_(IAudioExternalSynthesizer, IUnknown)
{
   STDMETHOD (QueryInterface) (THIS_ REFIID riid, LPVOID  *ppvObj) PURE;
   STDMETHOD_(ULONG,AddRef)   (THIS) PURE;
   STDMETHOD_(ULONG,Release)  (THIS) PURE;
};
typedef IAudioExternalSynthesizer *PIAUDIOEXTERNALSYNTHESIZER;


#endif    // _SPEECH_
```

E.44 StatsReportDlg.cpp

```
// StatsReportDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "StatsReportDlg.h"
```

```
    //{{AFX_DATA(CStatsReportDlg)
    enum { IDD = IDD_STATSREPORT };
    floatm_Homo;
    floatm_Hyp;
    floatm_PerCheck;
    floatm_Fitness;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CStatsReportDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CStatsReportDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_STATSREPORTDLG_H__62F67C21_A805_11D3_8082_0090276F4183__INCLUDED_)
```

E.46 TransformsDlg.cpp

```
// TransformsDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "TransformsDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CTransformsDlg dialog


CTransformsDlg::CTransformsDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CTransformsDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CTransformsDlg)
    m_XR = 0;
    m_XT = 0;
    m_YR = 0;
    m_YT = 0;
    m_ZR = 0;
    m_ZT = 0;
    //}}AFX_DATA_INIT
}


void CTransformsDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CTransformsDlg)
```

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

#endif // !defined(AFX_TRANSFORMSDLG_H__1363B73B_6C47_11D3_8079_0090276F4183__INCLUDED_)
```

E.48 ViewDataDlg.cpp

```
// ViewDataDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Isocurve.h"
#include "ViewDataDlg.h"

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

/////////////////////////////////////////////////////////////////////////////
// CViewDataDlg dialog


CViewDataDlg::CViewDataDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CViewDataDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CViewDataDlg)
    m_Dose = 0.0f;
    m_Type = _T("");
    m_X = 0;
    m_Y = 0;
    //}}AFX_DATA_INIT
}


void CViewDataDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CViewDataDlg)
    DDX_Text(pDX, IDC_DOSE, m_Dose);
    DDV_MinMaxFloat(pDX, m_Dose, 0.f, 1000000);
    DDX_Text(pDX, IDC_TYPE, m_Type);
    DDV_MaxChars(pDX, m_Type, 128);
    DDX_Text(pDX, IDC_X, m_X);
    DDV_MinMaxInt(pDX, m_X, 0, 1000);
    DDX_Text(pDX, IDC_Y, m_Y);
    DDV_MinMaxInt(pDX, m_Y, 0, 1000);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CViewDataDlg, CDialog)
    //{{AFX_MSG_MAP(CViewDataDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CViewDataDlg message handlers
```

E.49 ViewDataDlg.h

```
#if !defined(AFX_VIEWDATADLG_H__14C71CA3_56E1_11D3_BAB6_00104B9AC0DD__INCLUDED_)
#define AFX_VIEWDATADLG_H__14C71CA3_56E1_11D3_BAB6_00104B9AC0DD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ViewDataDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CViewDataDlg dialog

class CViewDataDlg : public CDialog
{
// Construction
public:
    CViewDataDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CViewDataDlg)
    enum { IDD = IDD_VIEWDATA };
    floatm_Dose;
    CStringm_Type;
    int     m_X;
    int     m_Y;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CViewDataDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CViewDataDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

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

#endif // !defined(AFX_VIEWDATADLG_H__14C71CA3_56E1_11D3_BAB6_00104B9AC0DD__INCLUDED_)
```