// CalcGap.cpp : implementation file

//

#include "stdafx.h"

#include "afxdialogex.h"

#include "CalcGap.h"

#include "PPMSDialog.h"

#include <string>

#include <vector>

#include <cmath>

#include <iterator>

#include <algorithm>

#include <iostream>

#include <fstream>

#include <iomanip>

#include <sstream>

using namespace std;

bool interpolate(vector<double> & vList, double dSeek, double & dResult);

/\*constants for calculating the gap \*/

const static double PI = 3.14159265359;

const static double EPSILON0 = 8.854187817; // in pF/m

const static CString PATH\_AGLIT = \_T("AgOB400.dat");

const static CString PATH\_CONFIG = \_T("DilConf.txt");

// CCalcGap dialog

IMPLEMENT\_DYNAMIC(CCalcGap, CDialog)

CCalcGap::CCalcGap(CString FilePath, CWnd\* pParent /\*=NULL\*/)

: CDialog(CCalcGap::IDD, pParent)

, m\_MeasFile(FilePath)

, m\_AgLitFile(PATH\_AGLIT)

, m\_TCOL(\_T("2"))

, m\_CCOL(\_T("3"))

, m\_Output(\_T(""))

, m\_OutputFile(\_T(""))

, m\_AgZero(\_T(""))

{

}

CCalcGap::~CCalcGap()

{

}

void CCalcGap::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Text(pDX, IDC\_EB\_MEASFILE, m\_MeasFile);

DDX\_Text(pDX, IDC\_EB\_AGLIT, m\_AgLitFile);

DDX\_Text(pDX, IDC\_EB\_C0, m\_C0);

DDX\_Text(pDX, IDC\_EB\_B, m\_b);

DDX\_Text(pDX, IDC\_EB\_RO, m\_ro);

DDX\_Text(pDX, IDC\_EB\_RI, m\_ri);

DDX\_Text(pDX, IDC\_EB\_TCOL, m\_TCOL);

DDX\_Text(pDX, IDC\_EB\_CCOL, m\_CCOL);

DDX\_Text(pDX, IDC\_OUTPUT, m\_Output);

DDX\_Text(pDX, IDC\_EB\_OUTPUTFILE, m\_OutputFile);

DDX\_Text(pDX, IDC\_EB\_AGZERO, m\_AgZero);

}

BEGIN\_MESSAGE\_MAP(CCalcGap, CDialog)

ON\_BN\_CLICKED(IDC\_BTN\_MEASFILE, &CCalcGap::OnBnClickedBtnMeasfile)

ON\_BN\_CLICKED(IDC\_BTN\_AGLIT, &CCalcGap::OnBnClickedBtnAglit)

ON\_BN\_CLICKED(IDC\_BTN\_CALCULATEGAP, &CCalcGap::OnBnClickedBtnCalculategap)

ON\_BN\_CLICKED(IDC\_BTN\_OUTPUTFILE, &CCalcGap::OnBnClickedBtnOutputfile)

ON\_BN\_CLICKED(IDC\_BTN\_HELP2, &CCalcGap::OnBnClickedBtnHelp2)

ON\_BN\_CLICKED(IDC\_BTN\_AGZERO, &CCalcGap::OnBnClickedBtnAgzero)

END\_MESSAGE\_MAP()

// CCalcGap message handlers

BOOL CCalcGap::OnInitDialog()

{

CDialog::OnInitDialog();

// TODO: Add extra initialization here

/\* Read dilatometer cell parameters from config file \*/

string line, strLabel, strValue;

ifstream ConfigFile(PATH\_CONFIG);

if ( ConfigFile.is\_open() )

{

while(getline(ConfigFile, line))

{

if(line[0] != '#' && line[0] != '{')

{

line.erase( remove( line.begin(), line.end(), ' ' ), line.end() ); //erase white spaces

unsigned pos = line.find("=");

strLabel= line.substr( 0, pos);

if(strLabel == "C0")

m\_C0 = line.substr(pos+1, line.find('#')-pos-1).c\_str();

else if(strLabel == "b")

m\_b = line.substr(pos+1, line.find('#')-pos-1).c\_str();

else if(strLabel == "ro")

m\_ro = line.substr(pos+1, line.find('#')-pos-1).c\_str();

else if(strLabel == "ri")

m\_ri = line.substr(pos+1, line.find('#')-pos-1).c\_str();

}

}

ConfigFile.close();

}

else

AfxMessageBox( \_T("Unable to read cell parameters from config file."), MB\_OK|MB\_ICONEXCLAMATION );

UpdateData(FALSE);

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

// EXCEPTION: OCX Property Pages should return FALSE

}

void CCalcGap::OnBnClickedBtnMeasfile()

{

// TODO: Add your control notification handler code here

CFileDialog MeasDlg(FALSE, \_T(".txt"), FALSE , OFN\_NOCHANGEDIR, \_T("Text Files (\*.txt)|\*.txt|") \_T("All Files|\*.\*||"));

if ( MeasDlg.DoModal() == IDOK )

m\_MeasFile = MeasDlg.GetPathName();

UpdateData(FALSE);

}

void CCalcGap::OnBnClickedBtnOutputfile()

{

// TODO: Add your control notification handler code here

CFileDialog MeasDlg(FALSE, \_T(".txt"), FALSE , OFN\_NOCHANGEDIR, \_T("Text Files (\*.txt)|\*.txt|") \_T("Data Files (\*.dat)|\*.dat|") \_T("All Files|\*.\*||"));

if ( MeasDlg.DoModal() == IDOK )

m\_OutputFile = MeasDlg.GetPathName();

// open file

ifstream OutputFile(m\_OutputFile);

if ( OutputFile.is\_open() )

{

/\* The file already exists. \*/

CString strMessage = \_T("The file ") + m\_OutputFile + \_T(" already exists.");

if ( AfxMessageBox(strMessage, MB\_OKCANCEL|MB\_ICONEXCLAMATION) == IDCANCEL)

m\_OutputFile = "";

}

UpdateData(FALSE);

}

void CCalcGap::OnBnClickedBtnAglit()

{

// TODO: Add your control notification handler code here

CFileDialog MeasDlg(FALSE, \_T(".dat"), \_T("AgOB400.dat"), OFN\_NOCHANGEDIR, \_T("Data Files (\*.dat)|\*.dat|") \_T("Text Files (\*.txt)|\*.txt|") \_T("All Files|\*.\*||"));

if ( MeasDlg.DoModal() == IDOK )

m\_AgLitFile = MeasDlg.GetPathName();

UpdateData(FALSE);

}

void CCalcGap::OnBnClickedBtnAgzero()

{

// TODO: Add your control notification handler code here

CFileDialog MeasDlg(FALSE, \_T(".dat"), \_T(""), OFN\_NOCHANGEDIR, \_T("Data Files (\*.dat)|\*.dat|") \_T("Text Files (\*.txt)|\*.txt|") \_T("All Files|\*.\*||"));

if ( MeasDlg.DoModal() == IDOK )

m\_AgZero = MeasDlg.GetPathName();

UpdateData(FALSE);

}

void CCalcGap::OnBnClickedBtnHelp2()

{

// TODO: Add your control notification handler code here

CString HelpText2 = \_T(" ");

ifstream helpfile("ReadMe2.txt");

if ( helpfile.is\_open() )

{

string line;

while (getline(helpfile, line) )

{

HelpText2 += line.c\_str();

HelpText2 += "\r\n";

}

helpfile.close();

}

else

{

HelpText2 += "Unable to open 'ReadMe2.txt'";

}

CHelpDialog dlgHelp(HelpText2);

dlgHelp.DoModal();

}

template <typename T> int sgn(T val)

/\* calulates signum \*/

{

return (T(0) < val) - (val < T(0));

}

void CCalcGap::OnBnClickedBtnCalculategap()

/\* Main Function

\* Calculates the gap from a measurement file (with capacitance and temperature columns)\*/

{

double C, T, dllAg, kT, AoT, AiT, C2, dstep, d, gamma2\_o, gamma2\_i, gamma\_o, gamma\_i;

int oldsign;

int ii, nsize;

string line;

stringstream ssMessage;

/\* load dialog entries \*/

UpdateData(TRUE); //important to update dialog variables (m\_AgLitFile, m\_TCOL, ...)

int nTcolumn = \_tstoi(m\_TCOL);

int nCcolumn = \_tstoi(m\_CCOL);

int nGAPcolumn = nCcolumn -1;

if (nGAPcolumn < 1 )

nGAPcolumn = 1;

//double dll, dllAgZero;

//double samplelength = atof(m\_L);

/\* load cell parameteres \*/

double C0 = \_tstof(m\_C0);

double b = 0.001 \* \_tstof(m\_b); // [mm] -> [m]

double ro = 0.001 \* \_tstof(m\_ro); // [mm] -> [m]

double ri = 0.001 \* \_tstof(m\_ri); // [mm] -> [m]

double k0 = EPSILON0/C0 \* (ro\*ro\*PI - ri\*ri\*PI); // [pF/m]/[pF] \* [m^2] = [m]

/\* load silver literature file into vector\*/

vector<double> vAgLit;

ifstream SilverLitFile(m\_AgLitFile);

if ( SilverLitFile.is\_open() )

{

//Problem of the old solution: skips the first line with numbers as well!

// ugly(?) solution: reset to beginning of file and read again

// get rid of header lines starting with # or {

int jj=0;

getline(SilverLitFile, line);

// count number of header lines

while (line[0] == '#' || line[0] == '{')

{

jj++;

getline(SilverLitFile, line);

}

SilverLitFile.seekg(0, SilverLitFile.beg);

for (ii=0; ii<jj; ii++)

getline(SilverLitFile, line);

copy(istream\_iterator<double>(SilverLitFile), istream\_iterator<double>(), back\_inserter(vAgLit));

SilverLitFile.close();

/\* normalise data to dll(T=300K)=0 \*/

double dllAg300 = 0;

if ( interpolate(vAgLit, 300, dllAg300) )

{

for (std::vector<double>::size\_type nn = 1; nn < vAgLit.size(); nn+=2)

vAgLit[nn]= vAgLit[nn] - dllAg300;

}

else

{

// silver.lit file is not going up to 300.

AfxMessageBox( \_T("Silver literature file out of range (value for T=300K needed) or inapropiate header lines (see help)."), MB\_OK|MB\_ICONEXCLAMATION );

return;

}

}

else

{

AfxMessageBox( \_T("Unable to open silver literature file."), MB\_OK|MB\_ICONEXCLAMATION );

return;

}

///\* load silver zero file into vector\*/

//vector<double> vAgZero;

//ifstream SilverZeroFile(m\_AgZero);

//if ( SilverZeroFile.is\_open() )

//{

// do { // get rid of header lines starting with # or {

// std::getline(SilverZeroFile, line);

// }while (line[0] == '#' || line[0] == '{');

// // Problem! This skips the first line with numbers as well!

// copy(istream\_iterator<double>(SilverZeroFile), istream\_iterator<double>(), back\_inserter(vAgZero));

// SilverZeroFile.close();

// /\* normalise data to dll(T=300K)=0 \*/

// double dllAgZero300 = 0;

// if ( interpolate(vAgZero, 300, dllAgZero300) )

// {

// // Dangerous! If vAgZero.size() is odd this will produce an error

// // Solution: < instead of !=, nn= 1 instead of 0, nn instead of nn+1

// for (std::vector<double>::size\_type nn = 1; nn < vAgZero.size(); nn+=2)

// vAgZero[nn]= vAgZero[nn] - dllAgZero300;

// }

// else

// {

// // silver.lit file is not going up to 300.

// AfxMessageBox( "Silver zero file out of range. Value for T = 300K needed.", MB\_OK|MB\_ICONEXCLAMATION );

// return;

// }

//}

//else

//{

// AfxMessageBox( "Unable to open silver zero file.", MB\_OK|MB\_ICONEXCLAMATION );

// return;

//}

/\* handle the measurement file \*/

//copyto << copyfrom.rdbuf();

if (m\_MeasFile == m\_OutputFile)

{

AfxMessageBox( \_T("Measurement and output file are the same."), MB\_OK|MB\_ICONEXCLAMATION );

return;

}

ifstream ReadFile(m\_MeasFile);

if ( !ReadFile.is\_open() )

{

AfxMessageBox( \_T("Unable to open measurement file."), MB\_OK|MB\_ICONEXCLAMATION );

return;

}

ofstream WriteFile(m\_OutputFile);

if ( !WriteFile.is\_open() )

{

AfxMessageBox( \_T("Unable to open output file."), MB\_OK|MB\_ICONEXCLAMATION );

return;

}

/\* handle headers \*/

int nHeaderLines = 1;

for (ii=0; ii<nHeaderLines; ii++)

{

getline(ReadFile, line);

stringstream ssline( line );

string field;

for (ii=0; ii<nGAPcolumn; ii++)

{

getline(ssline, field, ',');

WriteFile << field << ",";

}

WriteFile << " gap(mm),";

WriteFile << ssline.rdbuf() << endl;

}

WriteFile << "#The gap has been calculated with the following cell parameters: C0="<<C0<<" b="<<1000\*b<<" ro="<<1000\*ro<<" ri="<<1000\*ri<<endl;

nHeaderLines++;

//WriteFile << "k0=" << setprecision(19) << k0<< endl;

/\* main loop \*/

vector <double> vLine;

int nLineCounter = nHeaderLines;

bool bSkipLine;

int SkippedPoints = 0;

while(getline(ReadFile, line))

{

nLineCounter++;

bSkipLine = false;

/\* convert line into array of doubles \*/

vLine.clear();

stringstream ssline(line);

string field;

while(getline( ssline, field, ',' ))

{

stringstream fs( field );

double f = 0.0; // (default value is 0.0)

fs >> f;

vLine.push\_back( f );

}

/\* get T and C \*/

T = vLine[nTcolumn-1];

C = vLine[nCcolumn-1];

/\* interpolate dllAG \*/

if ( interpolate(vAgLit, T, dllAg) && C > 0.0 )

{

/\* calculate set of temperature dependent parameters \*/

kT = k0;// \* (1- dllAg);

AoT = ro\*ro\*PI\*(1+dllAg); // explain why no square

AiT = ri\*ri\*PI\*(1+dllAg);

C2 = 1000; //delete

dstep = 0.000001;

d = (AoT - AiT) \* EPSILON0/C;

if ( d < kT\*ro/(b+ro) )

d = 1.1 \* kT\*ro/(b+ro);

oldsign = 0;

ii=0;

while (abs(C - C2) > 1E-9 && dstep/d > 1E-15 && ii < 1000000)

{

ii++;

gamma\_o = ro/b \* (kT/d-1.);

gamma2\_o = gamma\_o\*gamma\_o;

gamma\_i = ri/b \* (kT/d-1.);

gamma2\_i = gamma\_i\*gamma\_i;

C2 = AoT\*EPSILON0/d \*2./gamma2\_o \* (1-sqrt(1-gamma2\_o));

C2 = C2 - AiT\*EPSILON0/d \*2./gamma2\_i \* (1.-sqrt(1.-gamma2\_i));

d = d + sgn(C2-C)\* dstep;

if (oldsign != sgn(C2-C))

{

oldsign = sgn(C2-C);

dstep = dstep/10;

}

}

if (ii == 1000000 || abs(C-C2) > 0.0001 )

{

ssMessage << "Iteration failed at line: " << nLineCounter << "\r\n";

bSkipLine = true;

SkippedPoints++;

}

}

else

{

if ( C <= 0.0 )

ssMessage << "Negative capacitance in line: " << nLineCounter << "\r\n";

else

ssMessage << "Temperature out of range. line: " << nLineCounter << "\r\n";

bSkipLine = true;

SkippedPoints++;

}

/\* calculate (Delta L)/L \*/

//if ( !bSkipLine )

//{

// /\* interpolate AgZero \*/

// if ( interpolate(vAgZero, T, dllAgZero))

// dll = d/samplelength - dllAgZero + dllAg; // I am not sure if dllAg is defined properly

//}

/\* write values to output file \*/

ii=0;

nsize = vLine.size();

if ( nGAPcolumn > nsize )

nGAPcolumn = nsize+1;

for ( ii = 0; ii<nGAPcolumn; ii++)

WriteFile << setw(11) << setprecision(9) << vLine[ii] << ", ";

if ( bSkipLine == true )

WriteFile << setw(14) << " ";

else

WriteFile << setw(14) << setprecision(12) << 1000\*d;

for (; ii<nsize; ii++)

WriteFile << ", " << setw(11) << setprecision(9) << vLine[ii];

WriteFile << endl;

}

ReadFile.close();

WriteFile.close();

ssMessage << "Calculation finished." << "\r\n";

ssMessage << "Skipped Datapoints: " << SkippedPoints << "\r\n";

//string strMessage;

//ssMessage >> strMessage;

m\_Output += ssMessage.str().c\_str();

UpdateData(false);

}