Final Year Project Report

Development of a C^{++} based user-interface for a plasma simulation tool

---

Author: Zhang, Junming (ID: 201138928)

Project Supervisor: Dr.Mark Bowden

Project Assessor: Dr.Kirsty McKay

Department of Electrical Engineering and Electronics

28 April 2017

Declaration of academic integrity

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated, falsified or embellished data when completing the attached piece of work. I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

SIGNATURE…… Junming Zhang ……

DATE …… 28 April 2017 ……

Abstract

A powerful microplasmas simulation tool Plasimo will generate many output data in various formats. In order to make this tool widely accessible to industry users and plasma researchers, a user-friendly interface is needed to develop.

The developed procedure includes background research, prototype design, program development and program improvement.

The project is software only and the implementation of this program based on C++ Visual Studio.

The Plasimo display assistant built in 20 weeks. It can be used to display each text file to the related propriety of plasma, generated 2D graphical images based on input data and created line chart based on the related text file.

Contents

1. Introduction

The objective of this project is to build a C^{++}-based user interface for helping Plasma researchers obtain the simulated result more efficient. This interface would serve as an accessory program to process output data from the plasma simulation tool Plasimo.

1.1 Background

Plasma is the fourth fundamental state of matter which consists of neutral ions, electrons, photos and ions.

Figure 1 illustrates the general type of plasma which has the vacuum chamber, pump, gas flow system and electrodes.

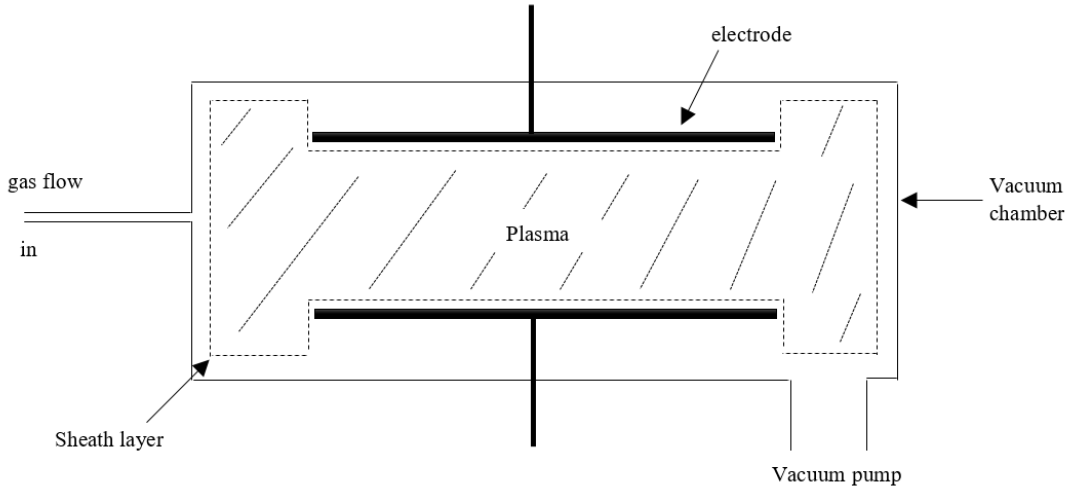


Figure 1. The general structure of plasma

Plasimo is a powerful program to provide the transient and steady-state simulation of microplasma in different dimensional geometries [1].

Figure 2 shows the main window of Plasimo 5.0 developer version. Users could load different models of plasma to run the simulation. At the same time, observe the changing process and obtain the simulated result.

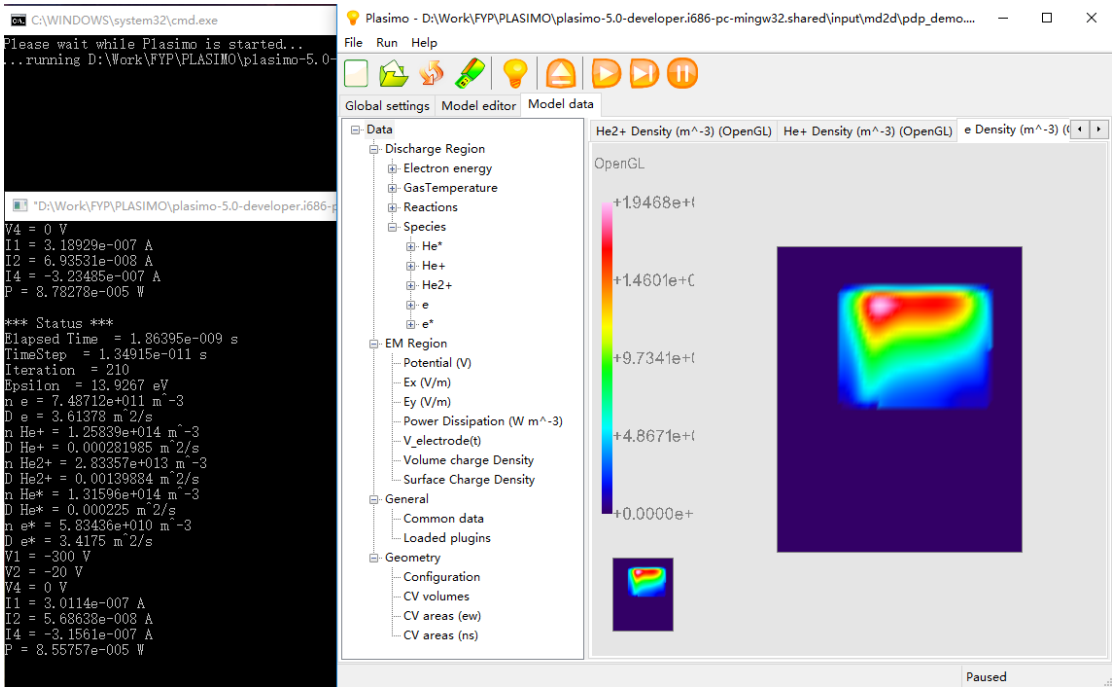


Figure 2. The main window of Plasimo developer 5.0 version

Supervisor Dr. Mark recommends the Micro Discharge 2D (pdp_md2d) for the target model in this project. It is a time- dependent model and the function of it is solving particle transport problem in conjunction [2].

Figure 3 displays the simulated result of the pdp_md2d model. Plasimo will generate 68 output files as “text” and “out” format in this case.

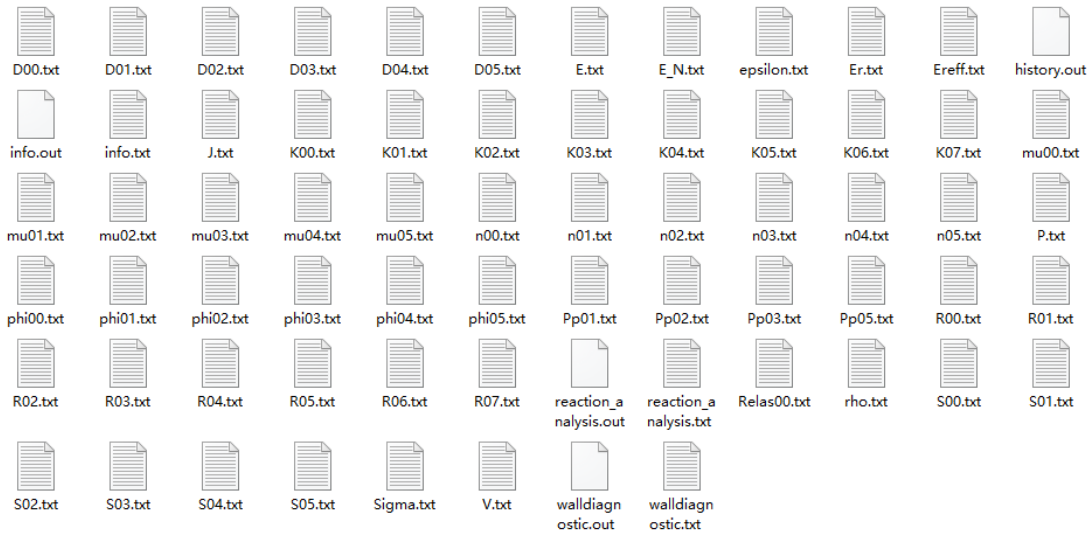


Figure 3. Simulated output files of pdp_md2d model

n00.txt	electron energy density [J m^{-3}]
phi00.txt	electron energy flux density [W m^{-2}]
S00.txt	electron energy source [W m^{-3}]
D00.txt	electron energy diffusion coefficient [W m^{2}]
mu00.txt	electron energy mobility coefficient [J m2 V^{-1} s^{-1}]
Relas00.txt	rate of electron energy loss from elastic collisions [W m^{-3}]
epsilon.txt	mean electron energy [J]
n01.txt	density for species 1 [m^{-3}]
S01.txt	source for species 1 [m^{-3} s^{-1}]
D01.txt	diffusion for species 1 [m^{2} s^{-1}]
mu01.txt	mobility for species 1 [m^{2} V^{-1} s^{-1}]
phi01.txt	flux for species 1 [m^{-2} s^{-1}]
R00.txt	reaction rate for reaction 1 [m^{-3} s^{-1}]
K00.txt	reaction rate coefficient for reaction 1 [m^{3} s^{-1}]
Pp01.txt	power dissipation for species 1 [W m^{-3}]
P.txt	dissipated power density [W m^{-3}]
J.txt	current density [C s^{-1} m^{-3}]
V.txt	potential [V]
E.txt	electric field [V m^{-1}]
Er.txt	reduced electric field E/p [V m^{-1} Pa^{-1}]
E_N.txt	reduced electric field E/N [V m^{2}]
rho.txt	volume charge density [C m^{-3}]
sigma.txt	surface charge density [C m^{-2}]
info.txt	the averaged values written with the user-specified frequency
info.out	the averaged values
history.out	gives the calculated variables as a function of time

Table 1. The corresponding properties of output files [3].

Table 1 illustrates the relationship of plasma properties and output files of Plasimo. This chart is obtained from the User Guild of Plasimo.

1.2 Problem statements

As mentioned in the background section, Plasimo provides a simulated program to investigate different plasma models. However, there are some drawbacks of Plasimo:

- The numerical output data saved in 68 different text files. Each one stores one specific propriety of plasma and their name is unreadable, users need the Table 1 in the Background section to find related proprieties.

- Figure 2 shows that users can only observe one property at one time and they cannot obtain the simulated time and numerical data in this dialog box. Therefore, it is inconvenient for Plasimo users to do further research.

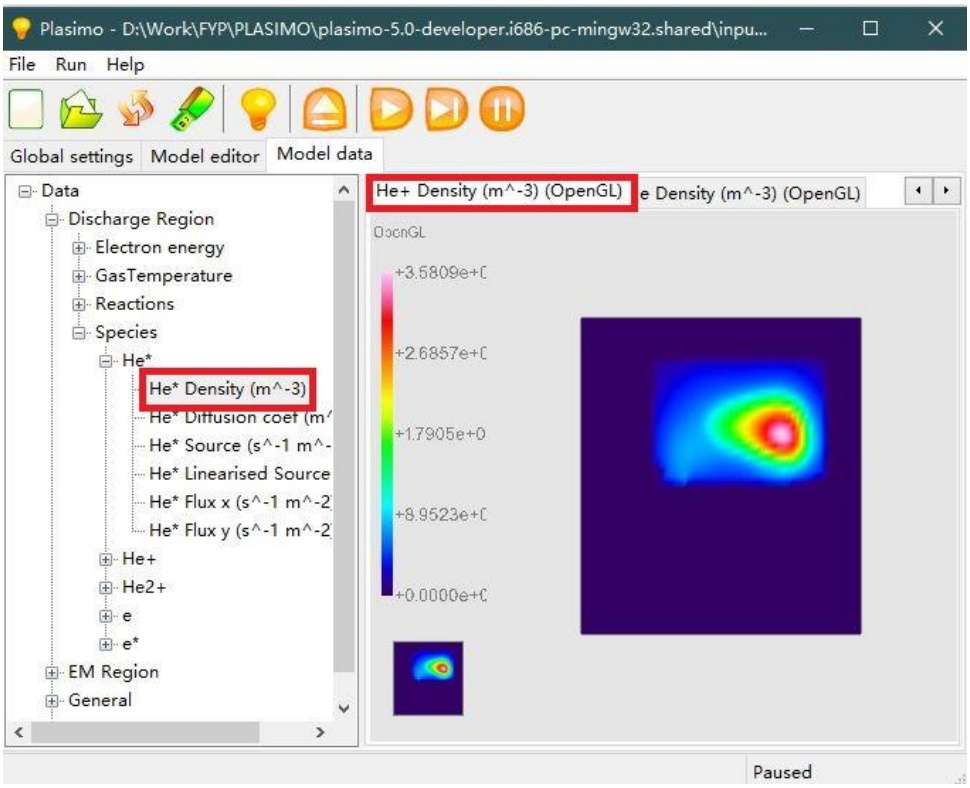


Figure 4. Simulation of pdp_md2d model

- The simulated process is irreversible and time-consuming. In this project, the pdp_md2d model usually cost 6 minutes to finish one simulation, if users want to

observe three properties, they need 18 minutes. Moreover, some other models need 1 day to do the simulation.

- A lot of output files generated through the simulation of plasma. In pdp_md2d model, there are 68 output files which mean 68 different properties of plasma. However, some of these properties are helpless to users.

In order to solve these drawbacks, a user-friendly interface is needed to develop. Hence, the objective of this project is to build a suitable user-interface to help researchers and industry users obtain required output data efficiently [4].

2. Literature Review [5]

6.1 The web-based user interface for EAST plasma control system

According to read the development part of EAST plasma control system, it could find that the interface design part is clearly because excellent workflows are used to express their ideas of function. Nonetheless, there still some drawbacks needed to avoid in this project by observing Figure 5, the intricate interface is unfriendly because this program needs to relieve time of users. Another is the narrow space between each button is unreadable.

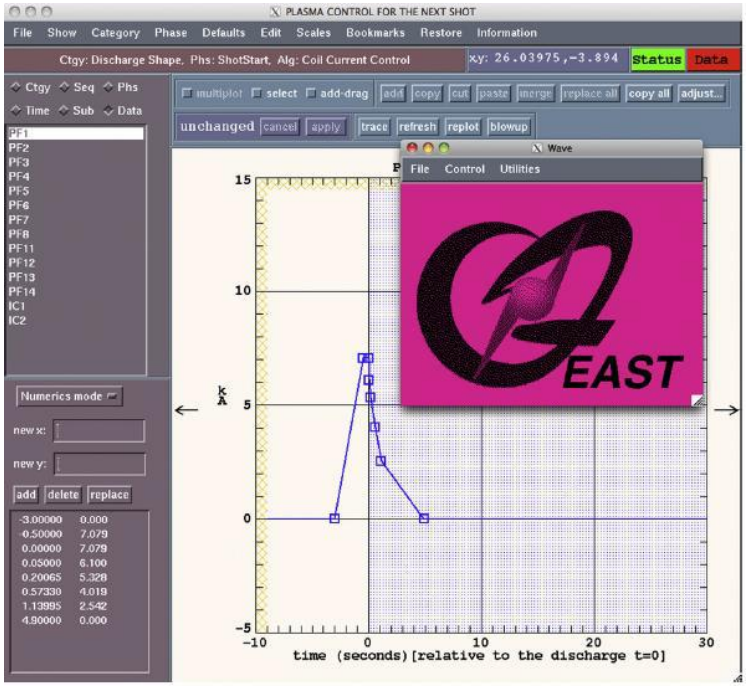


Figure 5. The user interface of EAST plasma control system

6.2 Web interface for plasma analysis codes

Figure 6 shows the interface of plasma analysis codes. All essential functions are display at one page include pictures, data, formulas and line charts. The concept of GUI design is worth to learn because users could observe the data when they do the simulation. However, all functions displayed on one page is an intricate design.

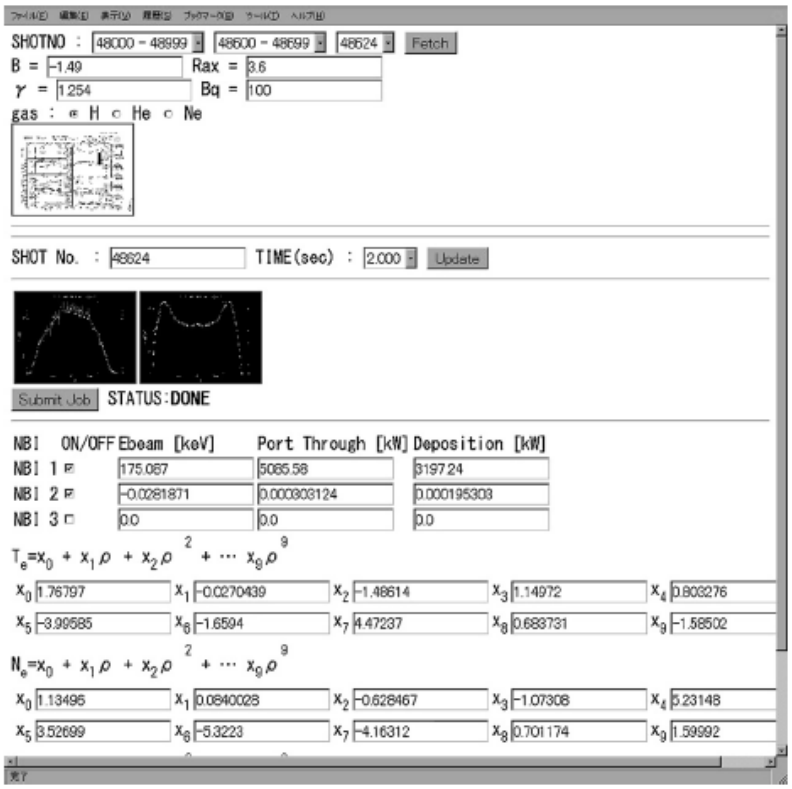


Figure 6. The Web interface for FIT code

6.3 User control interface for W7-X plasma operation

To reduce the workload of users is a significant concept from the design part of this monitoring system. The function of the program should meet the requirements of users and the program itself should finish tedious work. Therefore, the user survey and the investigate of related software is a major step before start design.

6.4 Literature reviews appendix

- R.R. Zhang, B.J. Xiao, Q.P. Yuan, F. Yang, Y. Zhang, R.D. Johnson, B.G. Penaflor, " The web-based user interface for EAST plasma control system", *J. Appl. Phys.* February 2014, DOI: 10.1016/j.fusengdes.2014.02.07016
- M. Emoto, S. Murakami, M. Yoshida, H. Funaba and Y. Nagayama., " Web interface for plasma analysis codes", *J. Appl. Phys.* vol. 83, no.2-3, pp. 453-457, April 2008, DOI: 10.1016/j.fusengdes.2007.10.008
- S. Anett, L. Heike, S. Jörg, " User control interface for W7-X plasma operation", *J. Appl. Phys.* 2007, DOI: 10.1016/j.fusengdes.2007.05.052

3. Theory

3.1 Developing flow path

Step 1: Investigating objective of the project.

Step 2: Exchange ideas with the supervisor.

Step 3: Analyse requirements of users.

Step 4: Propose essential functions of the program.

Step 5: Prototype design of program include interface and interaction.

Step 6: Slicing from prototype to build dialogs of the program.

Step 7: Program development based on the design.

Step 8: Program first demonstration and obtain suggestions.

Step 9: Program majorization and debug.

3.2 Project Plan

1) Preparatory Work	
1.1) Obtain relevant tool and materials	Week 4
2) Research Work	
2.1) Weekly background reading	Week 11

2.2) Literature review	Week 8
2.3) Learning Visual Studio, OpenGL	Week 16
3) Developing Work	
3.1) Writing software requirements	
3.1.1) Software Specifications	Week 8
3.1.2) Software Analysis	Week 9
3.2) Design	
3.2.1) Prototype design	Week 7
3.2.2) The main page developing	Week 7
3.2.3) Reading and Processing data function	Week 8
3.2.4) Menu developing	Week 9
3.2.5) Line chart function	Week 10
3.2.6) 2D graphic function	Week 16
3.3) Testing and Improving	Week 18
3.3.1) Program test and debug	Week 18
4) Presentation and Bench Inspection	
4.1) Presentation preparation	Week 12
4.2) Project video	Week 18
4.3) Bench inspection speech draft and poster	Week 19
5) Report Work	
5.1) Writing project specification form	Week 3
5.2) Writing preliminary report	Week 4
5.3) Weekly virtual log book	Week 18
5.4) Writing final report	Week 24

The complete project plan will show in Appendix 2 with the new Gantt chart.

4. Prototype Design

The prototype model is a significant step of design, the advantage of it is saving time in the program developing stage. It not just a single draft on the paper, it equals to a simulated program because the dialog and widget can be designed in this process. Moreover, the human-computer interaction and widget trigger were also built in this section. It supports an intuitionistic model to communicate with the supervisor. Therefore, the feedback can be given at the meeting and used to improve the design. Finally, the widget can be exported for real program development.

4.1 Tool

Mockplus is a practical prototype design tool. It is very easy to use because users can pull existed widgets to the mast page and set the respond command of them directly. Besides, this tool provides a bulk of elements and the designed model can be displayed on the iPad and mobile phone.

4.2 Design process

An interface should give users a comfortable experience and easy to use. In the design process, it obeys the Eight Golden Rules of Interface design [6].

1. Strive for consistency
2. Cater to universal usability
3. Offer informative feedback
4. Design dialogs to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

The purpose of this interface is to let users observe the output data from Plasimo efficiently. Therefore, the theme of this design is concise and it should reduce the operation of users to decrease the learning cost.

Homepage design:

The homepage is the first view of the program. New users are not familiar with it. Thus, a quick instruction should be displayed here and introduce users primary function and learn how to use it. Besides, there should have a help button to explain the outspread function and some common problems. The QR code is used to open the virtual log book of this project if users are interested in the developing process.

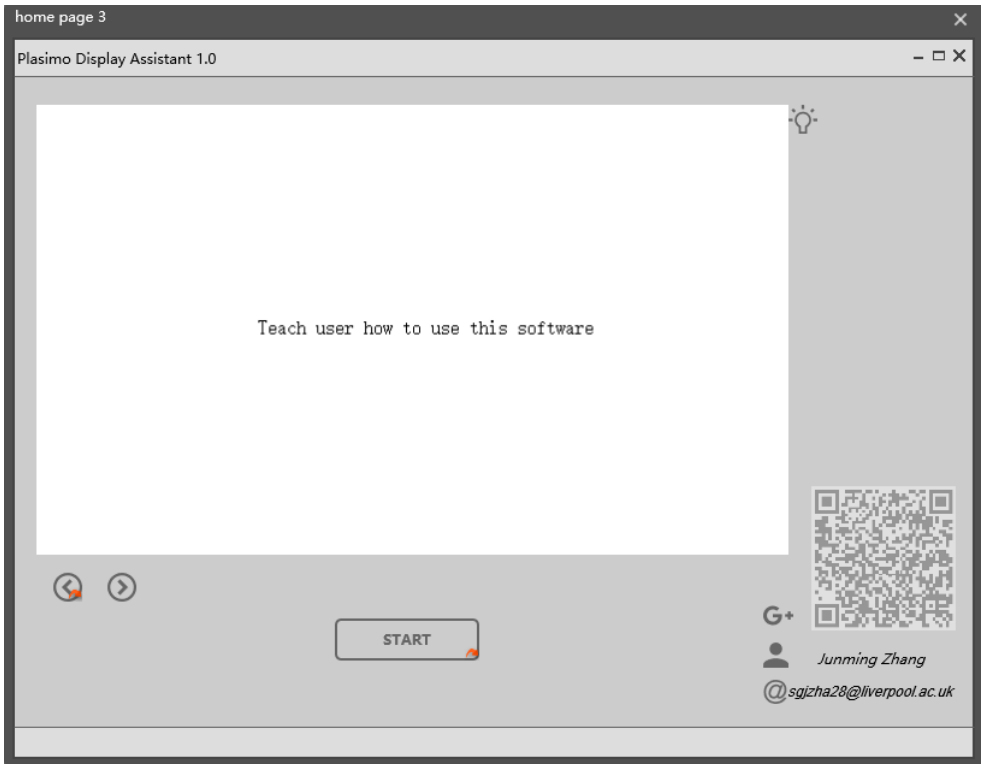


Figure 7. The home page of prototype design

Menu design:

The most basic plasma properties are [7]:

- Gas Density and pressure
- Neutral particle density
- Electron density
- Electron energy and electron temperature
- Ion density
- Plasma potential
- Ionization ratio

Lots properties of plasma bring a long menu. A custom menu can be used to solve this problem. Users could put the frequently-used features into the quick menu.

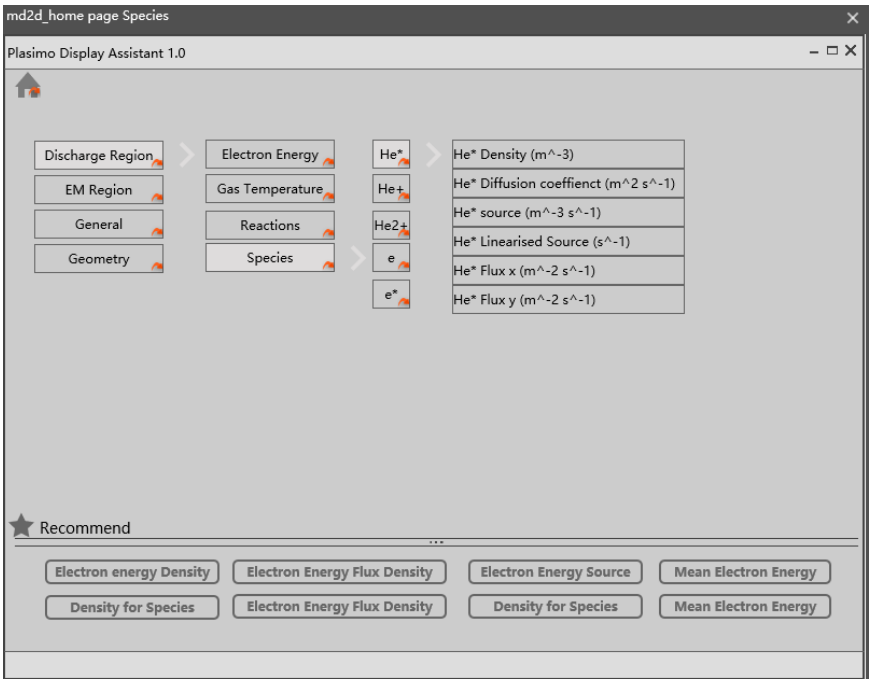


Figure 8. The menu of prototype design

Main window design:

According to the problem statement, users need to observe the changing process and numerical data at the same time. The main window should include these two part. One is the 2D graphical image based on output data. Another is the numerical data display.

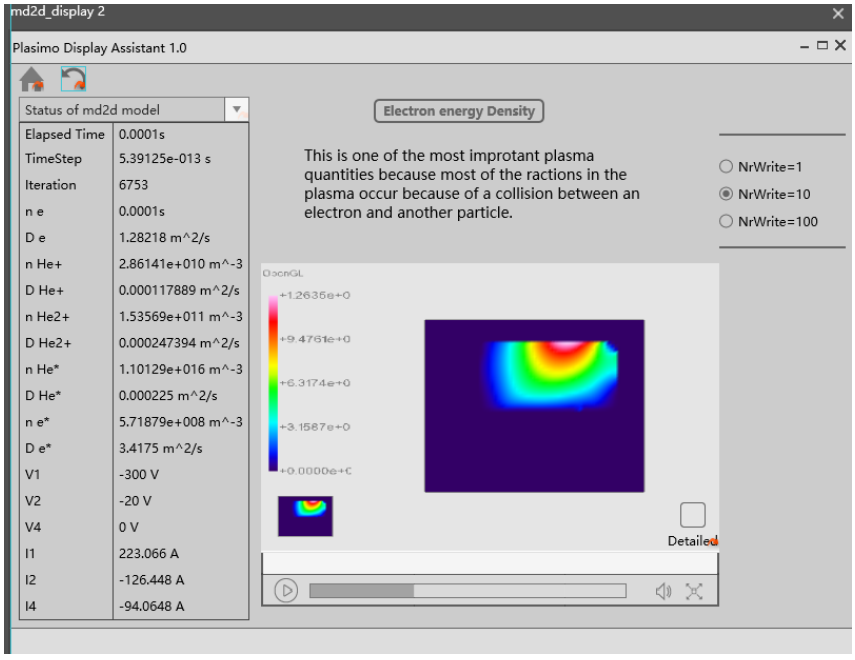


Figure 9. The main window of prototype design

5. Program Development

5.1 Tools

5.1.1 Microsoft Visual Studio Community 2015

Visual Studio is a primary program developing software which could apply on Windows platform. This project will use MFC in VS to build the frame of the program and use Visual Studio to develop the function of the interface.

5.1.2 OpenGL

Open Graphics Library (OpenGL) is the most widely 2D and 3D Application Program Interface (API). It could apply to lots of platforms such as Window, Linux and MacOS [8].

5.1.3 OpenCV

Open Source Computer Vision Library is a C++ vision development toolkit which contains basic image process function. For example, image copy, input and output.

5.2 Analysis

5.2.1 Software requirements

The purpose of this interface is pre-processing the output data from the Plasimo. This program should be used to solve the problems where stated in the problem statements section. Figure 10 is the feature summary of Plasimo and developing interface.

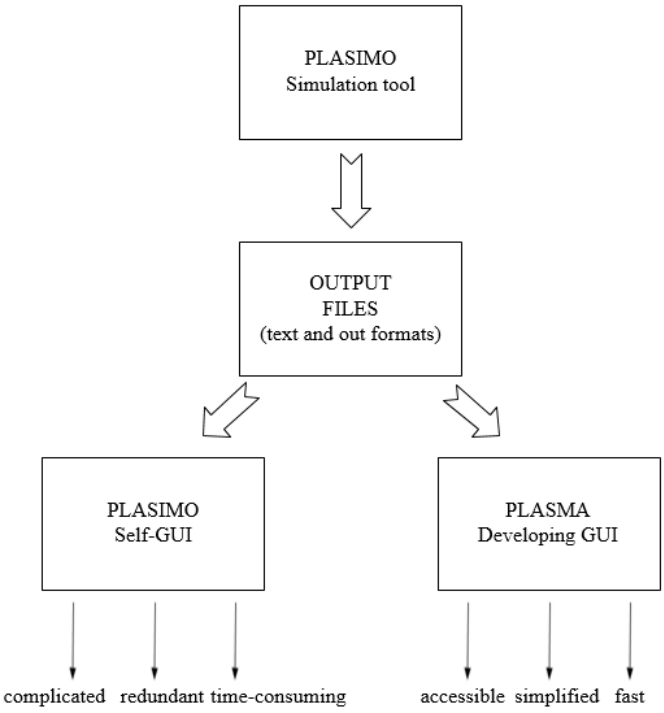


Figure 10. The summary of developing interface

5.2.2 Function specification

Homepage:

Based on the prototype design, response function of the widget is needed.

- Image loading function: display specific image on the widget.
- Page turning function: display different image of instruction.
- Help button function: Open the program specification pdf file.
- QR code: Link to Google+ blog.
- Start button function: Open the folder select box to load data.

Main Window:

The menu will change to pull-down menu to save the space of this program. This menu will be set on the top are of the main window.

- Text file display function: Locate specific text file to corresponding properties of plasma.
- Data loading function: Read data from particular text file.
- 2D graphical image display function: create the 2D graphical image based on the input data
- Menu response function: Display specific data when the user clicks the corresponding menu.

Line chart window:

Output data from the history.out is needed by Plasimo users usually. Thun, create the line chart based on it to help them.

- Properties matching function: Locate the specific data from history.out to corresponding properties.
- Generate the scale of axis function: The scale of x and y-axis should be changed to meet an appropriate interval.
- Line chart drawing function: Put each point on the coordinate system.

5.3 Function Realization

The basic procedure is draw the dialog of each window using MFC, and then write the response command to each widget in the dialog, finally combine all function together.

Dialog construction:

1. Create a new window in the Resource View.
2. Add the widget to the window based on the prototype.
3. Adjust the properties of each widget and change the title of the window.

5.3.1 Homepage

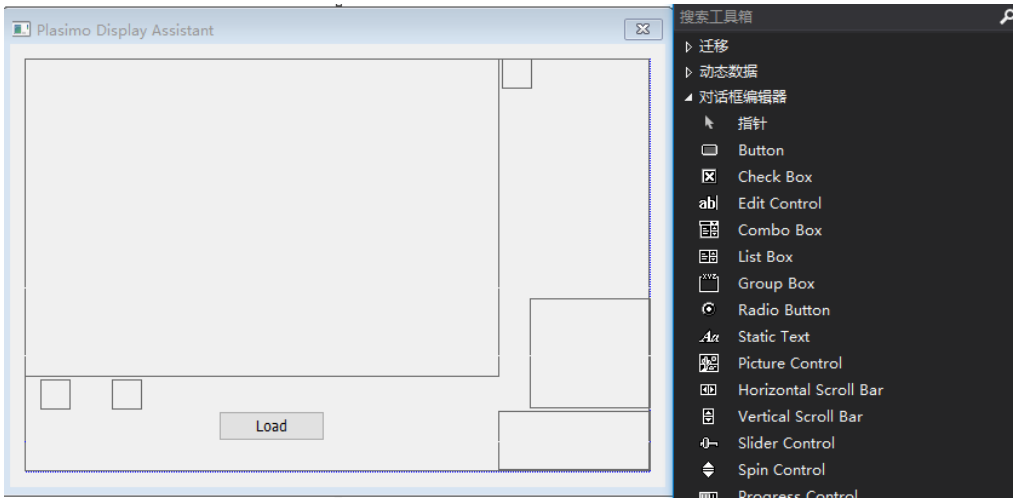


Figure 11. Build the dialog of homepage

Image loading function:

DrawPicToHDC is an OpenCV command and the function of it is display image to a particular widget. However, the image format in OpenCV should be cv::mat but the MFC cannot display this format directly, therefore the CvvImage.cpp file is used to transform cv::mat to the CvvImage format which can be shown in MFC. The CvvImage.cpp is a compensatory class of OpenCV.

```
void CFistPageMainDlg::DrawPicToHDC(char * pathfile, UINT ID)
{

    IplImage *image = NULL; //initial image
    if (image) cvReleaseImage(&image);

    image = cvLoadImage(pathfile, 1); //display image

    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC = pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
    cimg.CopyOf(image); // copy image
    cimg.DrawToHDC(hDC, &rect); // move this image to right widget area
    ReleaseDC(pDC);
}
```

Figure 12. This part code is used to add image to widget

In this project, the created images need to put in specific widgets in the dialog. Following code is used to display pictures of each widget on the homepage and the related image shown in Figure 14.

```
//Display initial iamges
void CFistPageMainDlg::InitFistPictureShow()//initialization
{
    DrawPicToHDC("externResouce\\FirstPicture\\Background.png", IDC_Pitcture_one);//background image
    DrawPicToHDC("externResouce\\FirstPicture\\91-100.png", IDC_PICTURE_2WEIMA);//QR code
    DrawPicToHDC("externResouce\\FirstPicture\\50-125.png", IDC_PICTURE_PERSONINFO);//my contact way
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-up.png", IDC_PICTURE_UP);//< button
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-down.png", IDC_PICTURE_DOWN);//> button
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-PDF.png", IDC_PICTURE_PDF);//help button

    DrawPicToHDC("externResouce\\FirstPicture\\FirstPicture.JPG", IDC_PICTURE_SHOW);//background image
}
void CFistPageMainDlg::ShowPicture(int num)
{
    switch (num)
    {

    case 0:
        DrawPicToHDC("externResouce\\FirstPicture\\FirstPicture.JPG", IDC_PICTURE_SHOW);//first image
        break;
    case 1:
        DrawPicToHDC("externResouce\\FirstPicture\\SecondPicture.JPG", IDC_PICTURE_SHOW);//second image
        break;
    case 2:
        DrawPicToHDC("externResouce\\FirstPicture\\ThirdPicture.JPG", IDC_PICTURE_SHOW);//third image
        break;
    default:

        break;
    }

}
```

Figure 13. The initialization of the homepage

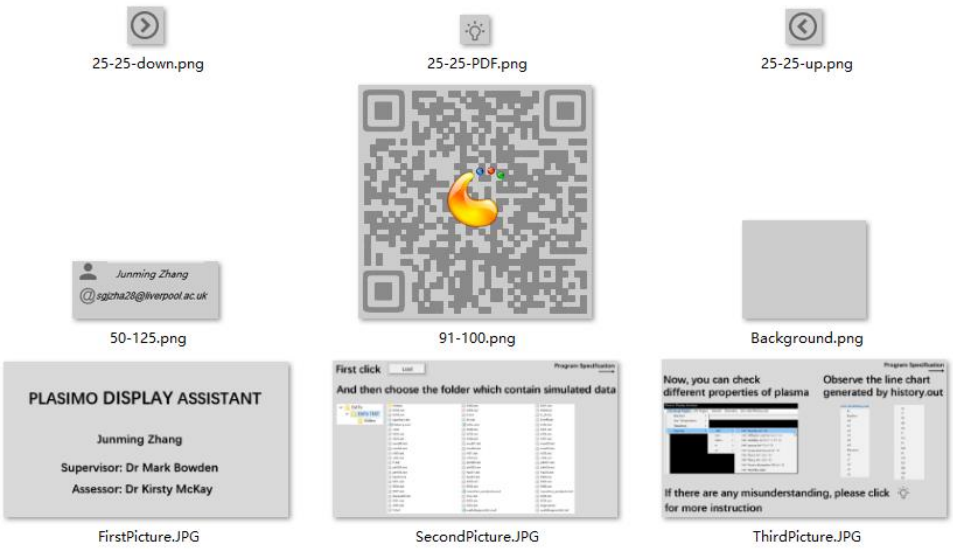


Figure 14. The image in the homepage

Click action:

```
//To judge the mouse area weather into the widget
bool CFistPageMainDlg::IsControlClick(CPoint point, UINT ID)
{
    CRect rect;
    GetDlgItem(ID)->GetWindowRect(&rect);
    ScreenToClient(rect);
    CPoint pt = (rect.TopLeft());

    GetDlgItem(ID)->GetClientRect(&rect);

    if (point.x > pt.x && point.x < (int)pt.x + rect.Width() &&
        point.y > pt.y && point.y < (int)pt.y + rect.Height()
        )
    {

        return true;
    }

    return  false;

}
```

Figure 15. the judge function of left clicking mouse

Figure 15 is the IsControlClick command, the function of it is to judge the left click action on the widget area. If users left click mouse, this command would return true to do following function:

- Page turning function: there are three instruction images controlled by clicking "<" and ">" buttons.
- Help button function: left-click this button to open the program specification.
- QR code: click this button will link to Google+ blog.

```
//Click action
void CFistPageMainDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值

    if (IsControlClick(point, IDC_PICTURE_2WEIMA))//click QR image
    {
          ShellExecute(NULL, _T("open"), _T("http://fypforjunmingzhang.blogspot.co.uk/"), NULL, NULL, SW_SHOW);
    }

    if (IsControlClick(point, IDC_PICTURE_UP))//click < button
    {



        if (--ID_SHOW_PICTURE <= 0)
            ID_SHOW_PICTURE = 0;
        ShowPicture(ID_SHOW_PICTURE);


    }
    if (IsControlClick(point, IDC_PICTURE_DOWN))//click > button
    {
        if (++ID_SHOW_PICTURE >= 2)
            ID_SHOW_PICTURE = 2;

        ShowPicture(ID_SHOW_PICTURE);

    }
    if (IsControlClick(point, IDC_PICTURE_PDF))//click help button
    {
                TCHAR szFilter[] = _T("文本文件(*.pdf)|*.pdf|");
                CFileDialog fileDlg(TRUE, _T("txt"), NULL, 0, szFilter, this);
                CString strFilePath;

                {
                    system("ProgramSpecification.pdf");
                    system("close");
                    // if click help button, it will onpen "programSpecification" file
                }
    }
    CDialogEx::OnLButtonDown(nFlags, point);//realize
}
```

Figure 16. the click action of page turning, help button and QR code

- Start button function: Open the folder select box and return the folder path which could link to next window.

```
//click"Load"
void CFistPageMainDlg::OnBnClickedButtonLoad()
{
    // TODO: 在此添加控件通知处理程序代码
    MyPropertySheet sh(_T("Plasimo Display Assistant"), this, 2); // Default display in third page
    CPage1 p1;

    // add one page
    sh.AddPage(&p1);



    CString szDir = OpenandChooseFileFolder();//Open infor.out
    if (szDir == "")
         return;
    CString infor_text = readInfoText(szDir);//Read infor.out
    if (infor_text == "")
        return;

    p1.m_editText = infor_text;
    sh.DoModal();              //information display

    //display the menu
    MenuDlg Menudlg;
    Menudlg.m_OpenFileFolder = szDir;
    Menudlg.DoModal();
}
```

Figure 17. the click action of load button

5.3.2 Main Window

The menu will connect to the dialog of the main window and it will display on the top area of it.

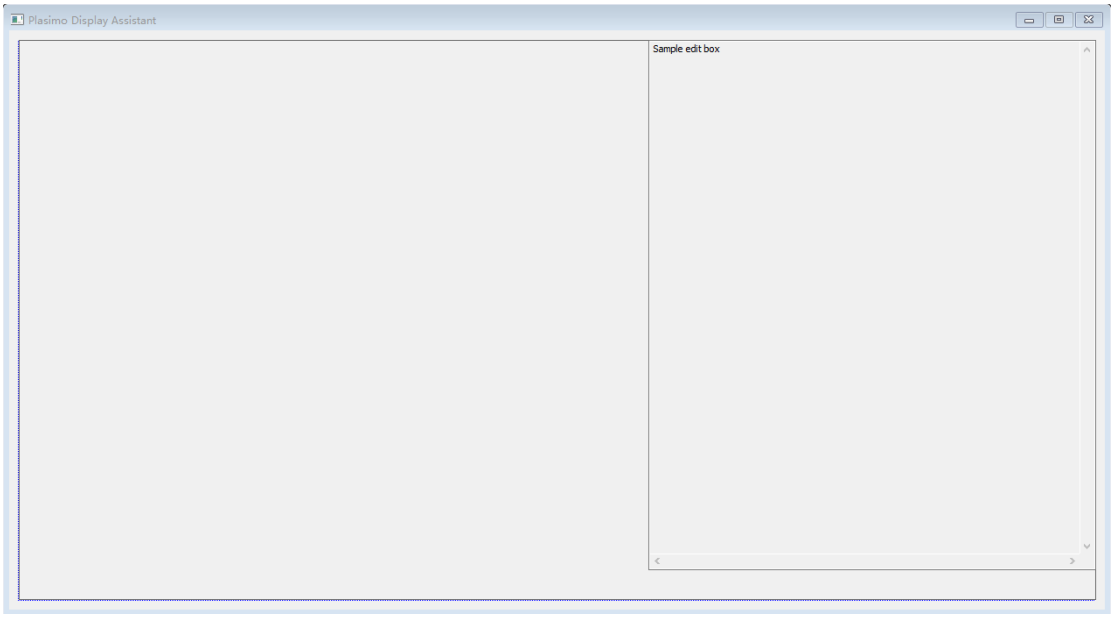


Figure 18. the dialog of main window

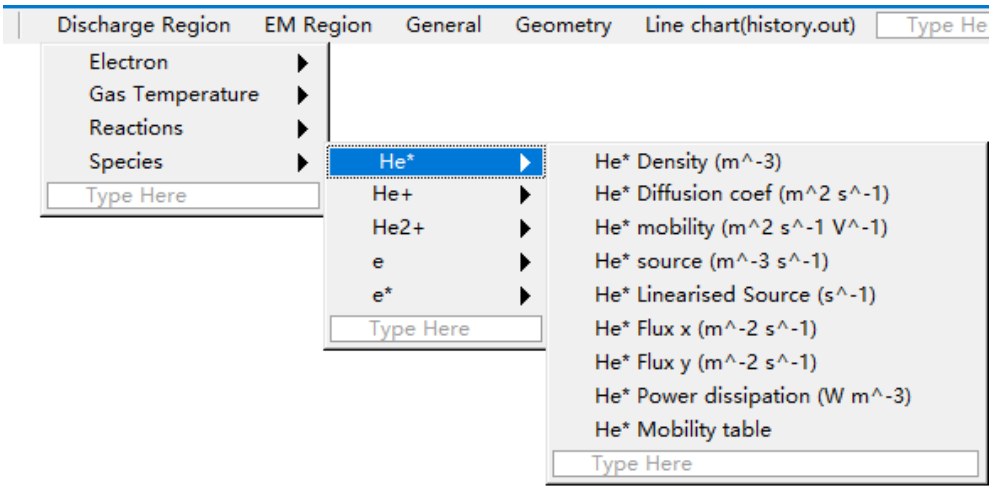


Figure 19. the menu of pdp_md2d model

Data loading function:

The objective of this program is pre-processing data from Plasimo. Therefore, the first step is load data function development. readInfoText command is used to obtain data in the reading model and put them into string m_EditShowTxt. In order to avoid memory wasting, this function will receive the length of data and then apply a memory based on the length.

```
CString  MenuDlg::readInfoText(CString filepath)//Read text file based on name
{

    if (!PathFileExists(filepath ))
        return "";


    CFile   file(filepath, CFile::modeRead);//open text file only for read
    char   *pBuf;
    int   iLen = file.GetLength(); //get the length of data
    pBuf = new   char[iLen + 1];
    file.Read(pBuf, iLen);
    pBuf[iLen] = 0;//add 0 at the end
    file.Close();

    CString A;
    A.Format("%s", pBuf);

    free(pBuf);
    m_EditShowTxt = A;
    UpdateData(FALSE);
    return  A;
}
```

Figure 20. the code of read text file function

In the previous step, the path load function in homepage will return the selected path. The final output information of selected model will display first in an information window.

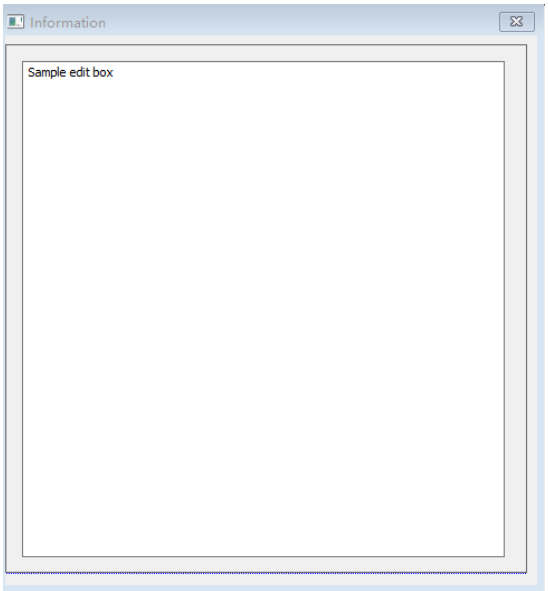


Figure 21. the dialog of information window

Menu Construction and text file display function:

Following is the matched result of each property in menu and output files. These results will be saved in a CSV file for data matching.

Discharge Region

Electron energy

Mean energy (J)	epsilon.txt
Mean energy density (J/m^3)	n00.txt
Elastic energy losses (W/m^3)	No match
Elastic energy linearised losses (1/m^3s)	No match
Mean energy source (W/m^3)	S00.txt
Mean energy linearised source (W)	No match
Mean energy flux density (x) (W/m^2)	phi00.txt #x-components
Mean energy flux density (y) (W/m^2)	phi00.txt #y-components
Reduced field (E/p)	Er.txt
Reduced field (E/N)	E_N.txt
Energy table	No match
Elastic loss table	No match

Table 2. The corresponding Electron energy properties of output files.

Gas Temperature

Gas temperature (K)	No match

Table 3. The corresponding Gas Temperature properties of output files.

Reactions

e + He <=> e + He* (Rate)	R00.txt
e + He <=> e + He* (Rate coefficient table)	K00.txt
e + He* <=> e + e + He^+ (Rate)	R01.txt
e + He* <=> e + e + He^+ (Rate coefficient table)	K01.txt
e + He <=> e + e + He^+ (Rate)	R02.txt
e + He <=> e + e + He^+ (Rate coefficient table)	K02.txt
He^+ + He + He <=> He^{2+} + He (Rate)	R03.txt
He^+ + He + He <=> He^{2+} + He (Rate coefficient table)	K03.txt
He* + He* <=> He^+ + He + e*(Rate)	R04.txt

He* + He* <=> He^+ + He + e*(Rate coefficient table)	K04.txt
He* + He* <=> He^{2+} + e*(Rate)	R05.txt
He* + He* <=> He^{2+} + e*(Rate coefficient table)	K05.txt
He* + e* <=> He + e*(Rate)	R06.txt
He* + e* <=> He + e*(Rate coefficient table)	K06.txt
He^{2+} + He <=> He^+ + He + He (Rate)	R07.txt
He^{2+} + He <=> He^+ + He + He (Rate coefficient table)	K07.txt

Table 4. The corresponding Reactions properties of output files.

Species

*He**

He* Density (m^-3)	n01.txt
He* Diffusion coef (m^2 s^-1)	D01.txt
He* mobility (m^2 s^-1 V^-1)	mu01.txt
He* source (m^-3 s^-1)	S01.txt
He* Linearised Source (s^-1)	No match
He* Flux x (m^-2 s^-1)	phi01.txt # x-components
He* Flux y (m^-2 s^-1)	phi01.txt #y-components
He* Power dissipation (W m^-3)	Pp01.txt
He* Mobility table	No match

Table 5. The corresponding He^* properties of output files.

He^+

He^+ Density (m^-3)	n02.txt
He^+ Diffusion coef (m^2 s^-1)	D02.txt
He^+ mobility (m^2 s^-1 V^-1)	mu02.txt
He^+ Source (m^-3 s^-1)	S02.txt
He^+ Linearised Source (s^-1)	No match
He^+ Flux x (m^-2 s^-1)	phi02.txt # x-components

He^+ Flux y (m^-2 s^-1)	phi02.txt #y-components
He^+ Power dissipation (W m^-3)	Pp02.txt
He^+ Mobility table	No match

Table 6. The corresponding He^+ properties of output files.

He^{2+}

He^{2+} Density (m^-3)	n03.txt
He^{2+} Diffusion coef (m^2 s^-1)	D03.txt
He^{2+} mobility (m^2 s^-1 V^-1)	mu03.txt
He^{2+} Source (m^-3 s^-1)	S03.txt
He^{2+} Linearised Source (s^-1)	No match
He^{2+} Flux x (m^-2 s^-1)	phi03.txt # x-components
He^{2+} Flux y (m^-2 s^-1)	phi03.txt #y-components
He^{2+} Power dissipation (W m^-3)	Pp03.txt
He^{2+} Mobility table	No match

Table 7. The corresponding He^{2+} properties of output files.

e

e Density (m^-3)	n04.txt
e Diffusion coef (m^2 s^-1)	D04.txt
e mobility (m^2 s^-1 V^-1)	mu04.txt
e Source (m^-3 s^-1)	S04.txt
e Linearised Source (s^-1)	No match
e Flux x (m^-2 s^-1)	phi04.txt # x-components
e Flux y (m^-2 s^-1)	phi04.txt #y-components
e Power dissipation (W m^-3)	Pp04.txt
e Mobility table	No match

Table 8. The corresponding e properties of output files.

*e**

e* Density (m^-3)	n05.txt

e* Diffusion coef (m^2 s^-1)	D05.txt
e* mobility (m^2 s^-1 V^-1)	mu05.txt
e* Source (m^-3 s^-1)	S05.txt
e* Linearised Source (s^-1)	No match
e* Flux x (m^-2 s^-1)	phi05.txt # x-components
e* Flux y (m^-2 s^-1)	phi05.txt #y-components
e* Power dissipation (W m^-3)	Pp05.txt
e* Mobility table	No match

Table 9. The corresponding e* properties of output files.

EM Region

Potential (V)	V.txt
Current density (C/m^3s)	J.txt
Ex (V/m)	E.txt # x-components
Ey (V/m)	E.txt # y-components
Power Dissipation (W m^-3)	P.txt
V_electrode(t)	No match
Volume charge Density	rho.txt
Surface Charge Density	Sigma.txt

Table 10. The corresponding EM Region properties of output files.

General

Common data	info.out

Table 11. The corresponding general properties of output files.

Geometry

Configuration	No match
CV volumes	No match
CV areas (ew)	No match
CV areas (ns)	No match

Table 12. The corresponding geometry properties of output files.

This program will read the matched result in CSV file and load them to a vector m_FileName. Therefore, a specific propriety can be located by related name and input path which is the Showfile command in Figure 25. Finally, the propriety can be displayed by putting the Showfile function to the response command of menu.

```
CString  ** MenuDlg::ReadCsvFile(CString filepath)//obtain the data name from the csv file
{

    CStringArray * Array = NULL;

    if (!PathFileExists(filepath))
        return NULL;
    CSVRead read(filepath);

    int Rowcount = read.FileRowCount();
    Array = read.CSVInf(' ');
    int Columcount = Array->GetSize();

    CString **p = new CString *[Rowcount];
    for (int i = 0; i < Rowcount; i++)
        p[i] = new CString[Columcount];
    for (int i = 0; i < Columcount; i++)
        p[0][i] = Array->GetAt(i);


    for (int i = 1; i<Rowcount; i++)
    {
        Array = read.CSVInf(' ');
        for (int j = 0; j < Columcount; j++)
            p[i][j] = Array->GetAt(j);

    }

    return p;

}
```

Figure 22. the code of reading csv file

```
m_FileName = CSVRead::ReadCsvFile("NameManagement\\DataName.csv" ,',',&row,&col);
```

Figure 23. the code of transfer csv data

```
void MenuDlg::OnElectronMeanenergy()
{

    ShowFile(0);
}
```

Figure 24. the code of matching data to menu

```
void   MenuDlg::ShowFile(int id)
{

    if (!PathFileExists(m_OpenFileFolder + "\\" + m_FileName[id][0])|| m_FileName[id][0]=="")// if not find the name of related text file
        m_EditShowTxt = "No Data Available";
    else
    {
        m_EditShowTxt = readInfoText(m_OpenFileFolder + "\\" + m_FileName[id][0]);
        if(id != 82)
        m_diaplay->Readtxt(m_OpenFileFolder + "\\" + m_FileName[id][0]);
    }
    UpdateData(FALSE);
```

Figure 25. the code of matching data

2D graphical image display function:

For generating the 2D image, the first step is divided the data into the independent number based on time. According to observe the output data, it could find that the symbol "#" separate the data in one period. Therefore, a loop is used in the code to read all data in one text file, when the "#" is detected, then store the present data in one vector colorpoints[]. Finally, this vector will record each data based on time.

```
while (!feof(fp))
{
    temp.clear();

    memset(szOneLine, 0, 1000);//Read one row
    fgets(szOneLine, 1000, fp);

    //ssObjFile.getline(szOneLine, 256);
    if (strlen(szOneLine) > 0)
    {
        HangVert vi;
        std::stringstream ssOneLine(szOneLine);

        if (szOneLine[0] != '#')//if not "#" it's the data
        {
            for (int i = 0; i < 60; i++)//obtain the data in each row
            {
                ssOneLine >> vi.colorpoint[i];
                if (vi.colorpoint[i] != 0 && vi.colorpoint[i] != 1)
                {
                    if (vi.colorpoint[i] > max[num])
                        max[num] = vi.colorpoint[i];
                    if(vi.colorpoint[i] < min[num])
                        min[num] = vi.colorpoint[i];
                }
            }
            colorpoints[num].push_back(vi);
        }
        else if(szOneLine[0] == '#'&&szOneLine[2] == 't')//if is "#" it mean it is the start of a series data of time
        {
            num++;
            //give a different color to different two time images
            for (int i = 0; i < 60; i++)
                vi.colorpoint[i] = 1010;
            colorpoints[num].push_back(vi);
            for (int i = 0; i < 60; i++)
                vi.colorpoint[i] = 1010;
            colorpoints[num].push_back(vi);

            //save the charater of time
            ssOneLine >> temp;
            ssOneLine >> temp;
            ssOneLine >> temp;
            ssOneLine >> temp;
            timestring[num].push_back(temp);
            timesheight[num].push_back(heigth);
```

Figure 26. the code of reading data from text file and processing them

For generating seven colors image, it needs to separate the data from one period to 7 regions. In this case, the maximum and minimum number will be found, and the difference value of them will be divided by 7, it could obtain seven numbers by using the minimum value to add the different multiples of this difference value. Finally, seven regions will be created and the interval of it will display as different numerical numbers.

```
float mid = (max[j] - min[j]) / 7.0;
float x1 = min[j] + mid;
float x2 = min[j] + 2 * mid;
float x3 = min[j] + 3 * mid;
float x4 = min[j] + 4 * mid;
float x5 = min[j] + 5 * mid;
float x6 = min[j] + 6 * mid;
float x7 = min[j] + 7 * mid;

CString s1;
s1.Format("Purple  :  %.2e -- %.2e", min, x1);
CString s2;
s2.Format("Blue    : %.2e -- %.2e", x1, x2);
CString s3;
s3.Format("cyan    : %.2e -- %.2e", x2, x3);
CString s4;
s4.Format("Green   : %.2e -- %.2e", x3, x4);
CString s5;
s5.Format("Yellow  : %.2e -- %.2e", x4, x5);
CString s6;
s6.Format("Orange  : %.2e -- %.2e", x5, x6);
CString s7;
s7.Format("Red    : :  %.2e -- %.2e", x6, x7);

float  midx = -200, midy = -mm_height +y ;

glColor3f(1, 0, 1);
glRasterPos3f(midx, -15+ midy, -250);
glListBase(mTexts - 32);
glCallLists(strlen(((std::string)s1.GetBuffer()).c_str()), GL_BYTE, ((std::string)s1.GetBuffer()).c_str());

glColor3f(0, 0, 1);
glRasterPos3f(midx, -30 + midy, -250);
glListBase(mTexts - 32);
glCallLists(strlen(((std::string)s2.GetBuffer()).c_str()), GL_BYTE, ((std::string)s2.GetBuffer()).c_str());
```

Figure 27. the code of spreading 7 regions in one period data

Different data will generate different regions. Therefore, the calculated areas transferred to integer 1 from 8. Following is the transformational formulas:

Suppose the 7 regions are x, 2x, 3x, 4x, 5x, 6x and 7x.

$$\text{output number} = \text{input number} \times \frac{7}{max - min} + (1 - \min \times \frac{7}{\max - min})$$

Hence,

$0 \leq \text{output number} < 1$ when $0 \leq \text{input number} < x$

$1 \leq \text{output number} < 2$ when $x \leq \text{input number} < 2x$

$2 \leq \text{output number} < 3$ when $2x \leq \text{input number} < 3x$

$3 \leq \text{output number} < 4$ when $3x \leq \text{input number} < 4x$

$4 \leq \text{output number} < 5$ when $4x \leq \text{input number} < 5x$

$5 \leq \text{output number} < 6$ when $5x \leq \text{input number} < 6x$

$6 \leq \text{output number} \leq 7$ when $6x \leq \text{input number} \leq 7x$

The 2D graphical image can be generated now by glcolor3f command.

0~1	purple	glcolor3f(1,0,1)
1~2	Blue	glcolor3f(0,0,1)
2~3	Cyan	glcolor3f(0,1,1)
3~4	Green	glcolor3f(0,1,0)
4~5	Yellow	glcolor3f(1,1,0)
5~6	Orange	glcolor3f(1,0.64,1)
6~7	Red	glcolor3f(1,0,0)

Table 13. The corresponding command of different colours

```
        double mid = 7 / (max[j] - min[j]);
        double addnum = 1 - min[j]*mid;
        glPushMatrix();
        glTranslatef(-30, 0, 0);
        glScaled(4, 4, 1);
        glEnable(GL_POINT_SMOOTH);
        glPointSize(6);
        glBegin(GL_POINTS);

        //Draw the image of data
        for (int i = 0; i < colorpoints[j].size(); i++)
        {


            for (int k = 0; k < 60; k++)
            {


                glColor3f(0.1, 0.2, 0.3);
                if (colorpoints[j][i].colorpoint[k] == 1010)
                    glColor3f(0.3, 0.2, 0.1);
                else if (colorpoints[j][i].colorpoint[k] != 0 && colorpoints[j][i].colorpoint[k] != 1)
                    setcolor(colorpoints[j][i].colorpoint[k] * mid + addnum); //transf the range to number 1~7
                glVertex2f(0.5*k, -0.5*i- height);

            }


        }
        height += 0.5*colorpoints[j].size();
        glEnd();
        glPopMatrix();
    }
}

glPopMatrix();
::glFinish();       // end of drawing
SwapBuffers( hdc);
wglMakeCurrent(hdc,NULL);
Sleep(1);
// Do not call CWnd::OnPaint() for painting messages
```

Figure 28. the code of processing data in 7 regions

```
void setcolor(double num)
{
    //Based on different value of data, drawing different colors

    int a = 1;
     if(num<=2)
    {

        glColor3f(1*a, 0, 1*a);
    }
    else if (num < 3 && num >= 2)
    {

        glColor3f(0, 0, 1 * a);

    }
    else if (num < 4 && num >= 3)
    {

        glColor3f(0, 1*a, 1 * a);

    }
    else if (num < 5 && num >= 4)
    {

        glColor3f(0, 1 * a, 0 );

    }
    else if (num < 6 && num >= 5)
    {

        glColor3f(1*a, 1 * a, 0);

    }
    else if (num < 7 && num >= 6)
    {

        glColor3f(1 * a, 0.64 * a, 0);
    }
    else if (num>= 7)
    {

        glColor3f(1 * a, 0, 0);
    }

}
```

Figure 29. the code of drawing the 2D graphical image

In order to browse all generated 2D graphical image, a mouse wheel with adjustable speed is used to control it.

```
BOOL CMyOpenGLView1::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    // TODO: Add your message handler code here and/or call default

    y -= 0.5*zDelta; // move speed of mouse wheel

    return CWnd::OnMouseWheel(nFlags, zDelta, pt);
}
```

Figure 30. the code of controlling 2D graphical image by mouse wheel

5.3.3 Line chart window

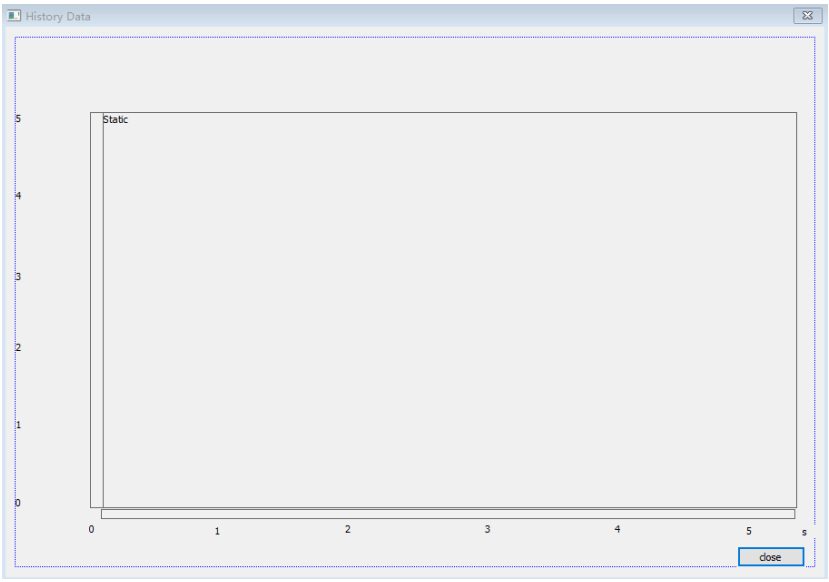


Figure 31. the dialog of line chart function

The first step is adding the unit to the line chart. And then read the data from history.out, the data is different in each property. Therefore the value of x and y-axis need change with the input data. The method has used a loop to find the maximum and minimum value of one column. After that, the difference value of them will be divided by 5 and five numbers can be calculated by using the minimum value to add the different multiples of the difference value. Finally, the scale of coordinate system can be set with corresponding intervals.

```
void DialogName(DialogGl * m_dialoggl, int biaohao)
{
    switch (biaohao)
    {
    case 1:
        m_dialoggl->m_glName = "dt";
        m_dialoggl->m_name = "S";
        break;
    case 2:
        m_dialoggl->m_glName = "Er";
        m_dialoggl->m_name = "Vm-1*Pa-1";
        break;
    case 3:
        m_dialoggl->m_glName = "Epsilon";
        m_dialoggl->m_name = "J";
        break;
    case 4:
        m_dialoggl->m_glName = "no";
        m_dialoggl->m_name = "J ";
        break;
    case 5:
        m_dialoggl->m_glName = "n1";
        m_dialoggl->m_name = " J";
        break;
```

Figure 32. the code of adding units

```
float*  CMyOpenGLView::ShowLine(int biaohao,float *kedu)
{
    biaohao_show = biaohao;
    minx = -1; miny = -1; maxx = -1; maxy = -1;

    if (data == NULL)
        return kedu;

        for(int i = 1; i < data_row; i++)
        {
            if (minx == -1)
            minx = atof(data[i][0]);
            if (maxx == -1)
            maxx = atof(data[i][0]);
            if (miny == -1)
            miny = atof(data[i][biaohao]);
            if (maxy == -1)
            maxy = atof(data[i][biaohao]);

            CString a = data[i][0];
            CString b = data[i][biaohao];

            minx = atof(data[i][0]) < minx ? atof(data[i][0]) : minx;
            maxx = atof(data[i][0]) > maxx ? atof(data[i][0]) : maxx;

            miny = atof(data[i][biaohao]) < miny ? atof(data[i][biaohao]) : miny;
            maxy = atof(data[i][biaohao]) > maxy ? atof(data[i][biaohao]) : maxy;

        }

        kedu[0] = minx;
        kedu[1] = maxx;
        kedu[2] = miny;
        kedu[3] = maxy;

        SetTimer(TIMERID, 10, 0);
        return kedu;
}
```

Figure 33. the code of processing data from history.out

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



    DrawPicToHDC("externResouce\\FirstPicture\\x刻度.png", IDC_PICTURE_X);
    DrawPicToHDC("externResouce\\FirstPicture\\y刻度.png", IDC_PICTURE_Y);


        kedu_x1.Format("%.2e", kedu[0]);
        kedu_x2.Format("%.2e", kedu[0] + (kedu[1] - kedu[0]) / 5);
        kedu_x3.Format("%.2e", kedu[0] + 2 * (kedu[1] - kedu[0]) / 5);
        kedu_x4.Format("%.2e", kedu[0] + 3 * (kedu[1] - kedu[0]) / 5);
        kedu_x5.Format("%.2e", kedu[0] + 4 * (kedu[1] - kedu[0]) / 5);
        kedu_x6.Format("%.2e", kedu[0] + 5 * (kedu[1] - kedu[0]) / 5);
                                                    // Divide the number to 6 areas to mark the scale
        kedu_y1.Format("%.2e", kedu[2]);
        kedu_y2.Format("%.2e", kedu[2] + (kedu[3] - kedu[2]) / 5);
        kedu_y3.Format("%.2e", kedu[2] + 2 * (kedu[3] - kedu[2]) / 5);
        kedu_y4.Format("%.2e", kedu[2] + 3 * (kedu[3] - kedu[2]) / 5);
        kedu_y5.Format("%.2e", kedu[2] + 4 * (kedu[3] - kedu[2]) / 5);
        kedu_y6.Format("%.2e", kedu[2] + 5 * (kedu[3] - kedu[2]) / 5);

        UpdateData(FALSE);

        int a = 0;
}
```

Figure 34. the code of setting coordinate system

The next step is using openGL to draw all points to the coordinate system, the value of x-axis will be the first column of history.out which is the time, the value of y-axis will

be other columns of history.out. Following is the formula of coordinates of x and y-axis:

$$\text{x cordinate} = \frac{\text{data[i][0]}}{(\max[x] - \min[x])} \times \text{width}$$

$$\text{y cordinate} = \frac{\text{data[i][Other]}}{(\max[y] - \min[y])} \times \text{hight}$$

```
void CMyOpenGLView::OnPaint() //Darw curve
{
    CPaintDC dc(this); // device context for painting
    wglMakeCurrent(hdc,hglrc);
    ::glClearColor(0.0f,0.0f,0.0f,1.0f);
    // TODO: Add your message handler code here
    ::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );//Clear Buffer

    glPushMatrix();//save the matrix
    glTranslatef(-gl_width / 2+1, -gl_heigth/2+1, 0);//location transf
    GLfloat curSizeLine = 1;
    glPointSize(curSizeLine);
    if (biaohao_show <= 1)//The first colume is time
        return ;
    glBegin(GL_POINTS);
    glColor3f(1.0f, 0.0f, 0.0f);

    for (int i = 1; i < data_row; i++)
    {

        glVertex2f(atof(data[i][0]) / (maxx - minx)*gl_width, atof(data[i][biaohao_show]) / (maxy - miny)*gl_heigth);//draw all points

    }

    glEnd();
    glPopMatrix();
    ::glFinish();       // End draw
    SwapBuffers( hdc);
    wglMakeCurrent(hdc,NULL);
    Sleep(1);
    // Do not call CWnd::OnPaint() for painting messages
}
```

Figure 35. the code of drawing line chart

```
void MenuDlg::OnLinechartEr()
{
    ShowLine(2);
}
```

Figure 36. the code of display specified line chart

6. Results

6.1 Interface installation

OpenCV configuration is the first step of installation. The program cannot build successfully without it because the image command is developed based on OpenCV.

Step 1:

Open the opencv-2.4.10.exe and release the OpenCV files to hard disk D which is shown in Figure 37.

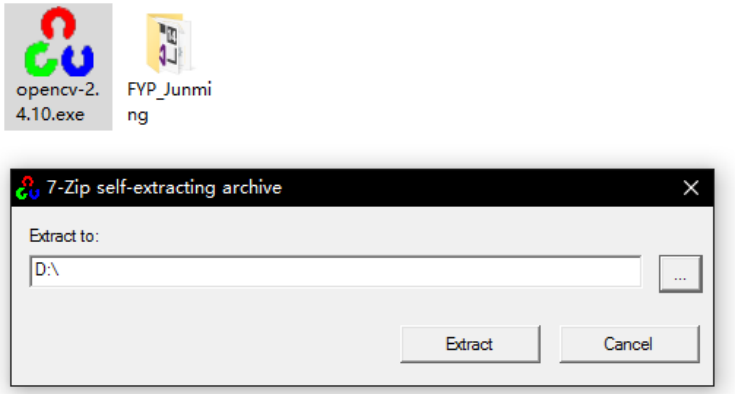


Figure 37. The configuration step 1 of OpenCV

Step 2:

Go to hard disk D, open the "bin" folder followed by clicking D:\opencv\build\x64\vc10\bin. Finally copy all files in this folder to system folder at C:\Windows\System32 and C:\Windows\ SysWOW64 which is shown in Figure 38.

Figure 38. The configuration step 2 of OpenCV

Step 3:

After that, open the project file “Junming_Zhang_FYP.sln” by using Visual Studio 2015 which is shown in Figure 39.

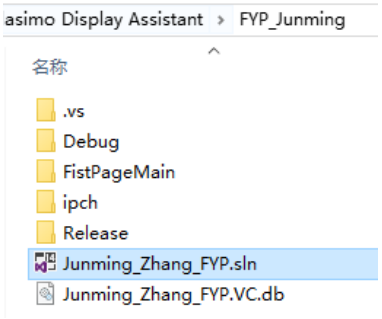


Figure 39. The configuration step 3 of OpenCV

Step 4:

Right-click the project name and then open the properties setting which is shown in Figure 40.

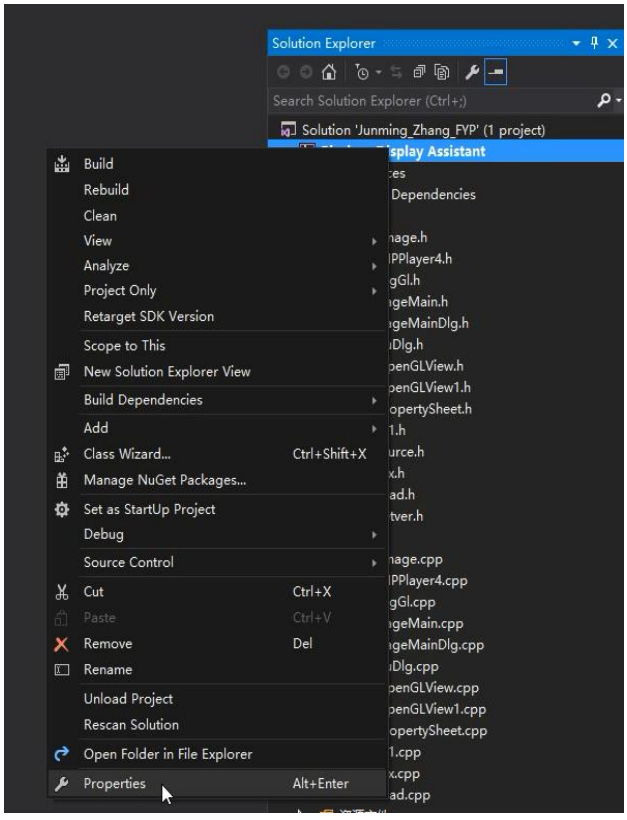


Figure 40. The configuration step 4 of OpenCV

Step 5:

Find the Include Directories of VC++ Directories in the Configuration Properties and then type the following address into it which is shown in Figure 41 and 42.

D:\opencv\build\include\opencv

D:\opencv\build\include\opencv2

D:\opencv\build\include

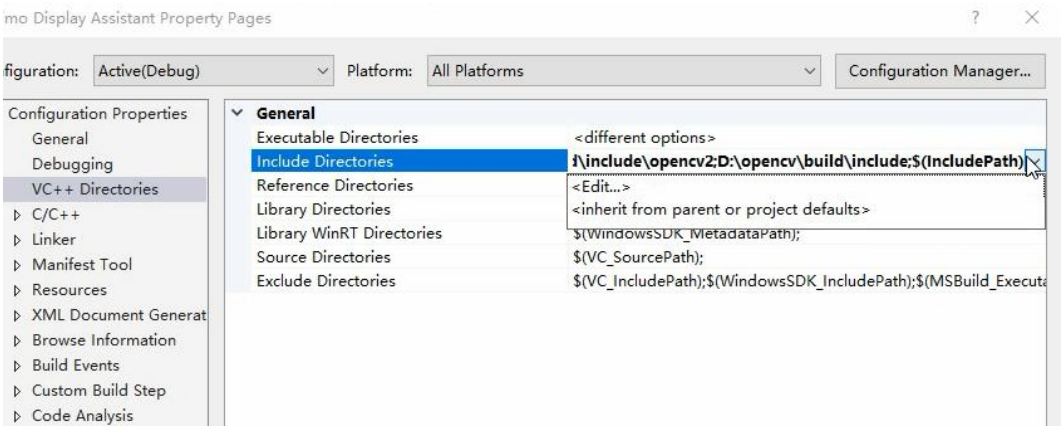


Figure 41. The configuration step 5 of OpenCV

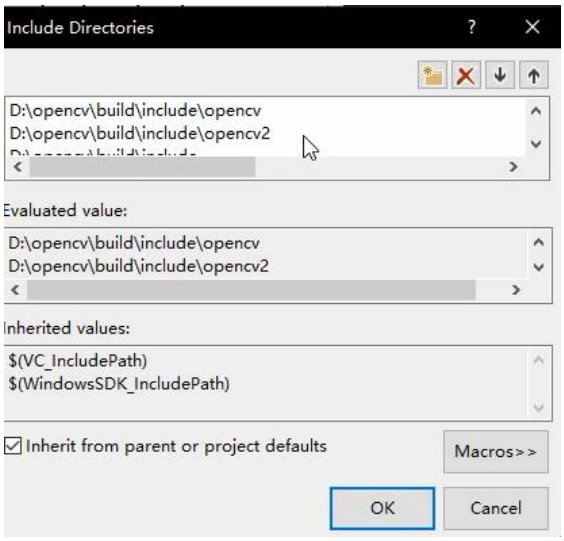


Figure 42. The configuration step 5 of OpenCV

Step 6:

It is similar to the previous step, find the Library Directories of VC++ Directories in the Configuration Properties and then type the following address into it which is shown in Figure 43 and 44.

D:\opencv\build\x86\vc10\lib

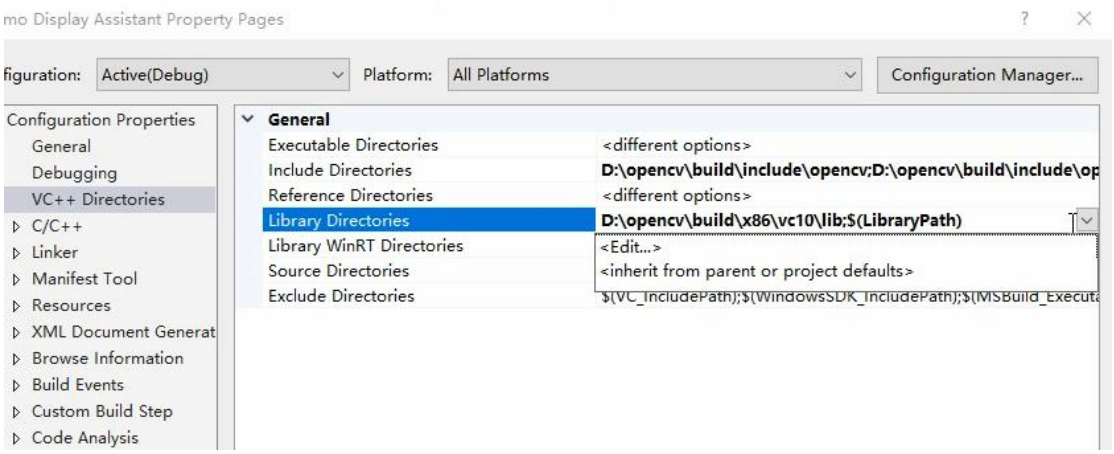


Figure 43. The configuration step 6 of OpenCV

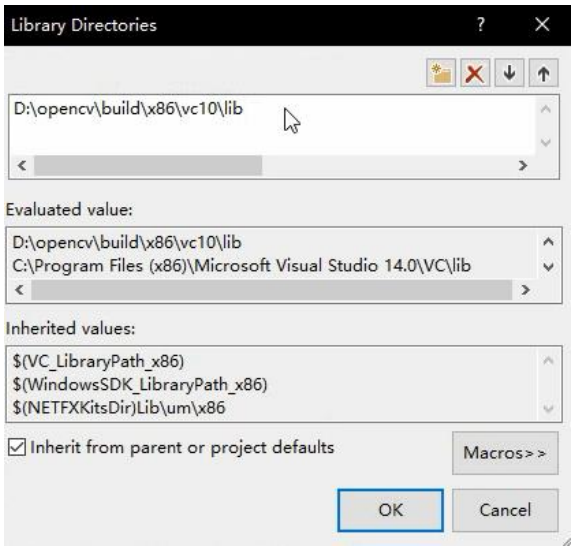


Figure 44. The configuration step 6 of OpenCV

Step 7:

The last step is to find the Additional Dependencies of Input in the Linker and then type the following names of library file into it which is shown in Figure 45 and 46.

opencv_ml2410d.lib
opencv_calib3d2410d.lib
opencv_contrib2410d.lib
opencv_core2410d.lib
opencv_features2d2410d.lib
opencv_flann2410d.lib
opencv_gpu2410d.lib
opencv_highgui2410d.lib
opencv_imgproc2410d.lib
opencv_legacy2410d.lib

opencv_objdetect2410d.lib
opencv_ts2410d.lib
opencv_video2410d.lib
opencv_nonfree2410d.lib
opencv_ocl2410d.lib
opencv_photo2410d.lib
opencv_stitching2410d.lib
opencv_superres2410d.lib
opencv_videostab2410d.lib
opencv_objdetect2410.lib
opencv_ts2410.lib
opencv_video2410.lib
opencv_nonfree2410.lib
opencv_ocl2410.lib
opencv_photo2410.lib
opencv_stitching2410.lib
opencv_superres2410.lib
opencv_videostab2410.lib
opencv_calib3d2410.lib
opencv_contrib2410.lib
opencv_core2410.lib
opencv_features2d2410.lib
opencv_flann2410.lib
opencv_gpu2410.lib
opencv_highgui2410.lib
opencv_imgproc2410.lib
opencv_legacy2410.lib
opencv_ml2410.lib

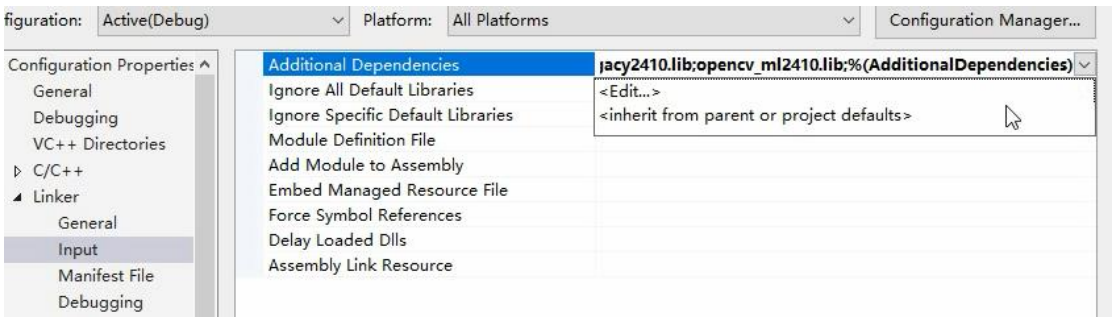


Figure 45. The configuration step 7 of OpenCV

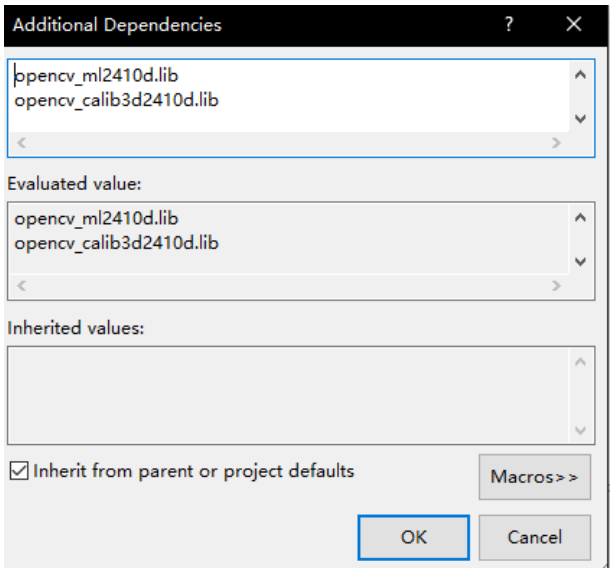


Figure 46. The configuration step 7 of OpenCV

The program can be built now by Visual Studio 2015. Click Local Windows Debugger to obtain the executable program which is shown in Figure 47.

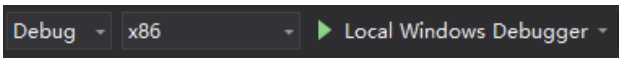


Figure 47. Build process of program

6.2 Interface demonstration

The program can be opened after build in the Visual Studio 2015. The location of the executive program is the debug folder which is shown in Figure 48.

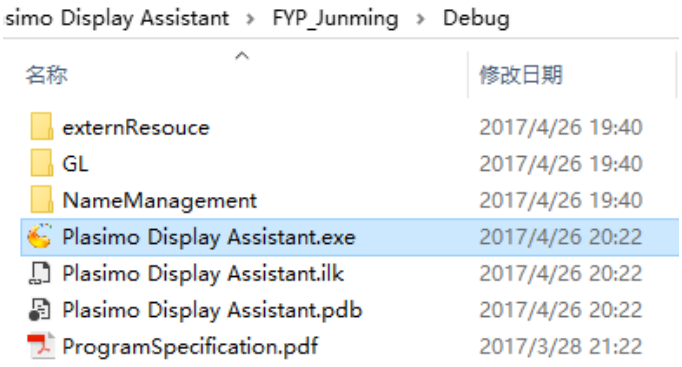


Figure 48. The executive program of this project

Homepage:

The color of the title will keep the same with the computer theme. This little function lets the program looking more appropriate on the desktop.

The first page displays the information of author, supervisor and assessor. Users could learn how to use the program by clicking the page turning button which is shown in Figure 49, 50 and 51.

The bulb symbol is help button. It will open the program specification to teach the user how to process regular problems and do the modification of these programs.

The QR code will link to the Google+ virtual logbook of this program. All developing process will upload to the blog after submitting this report.

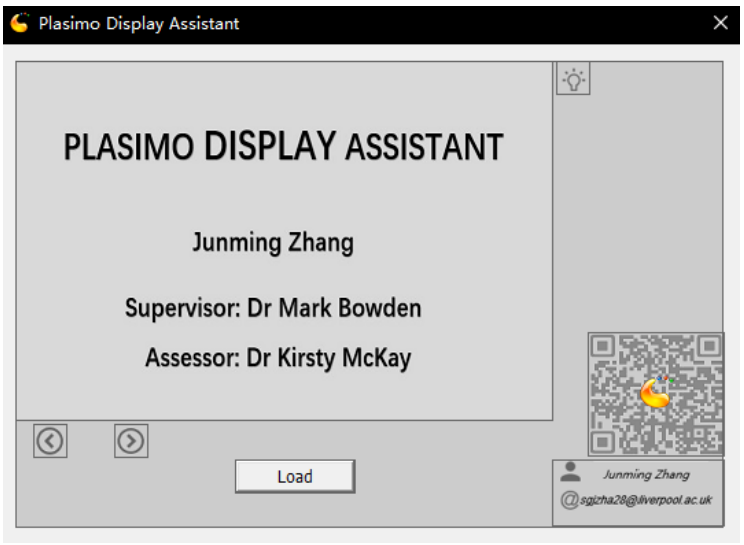


Figure 49. The first page of homepage

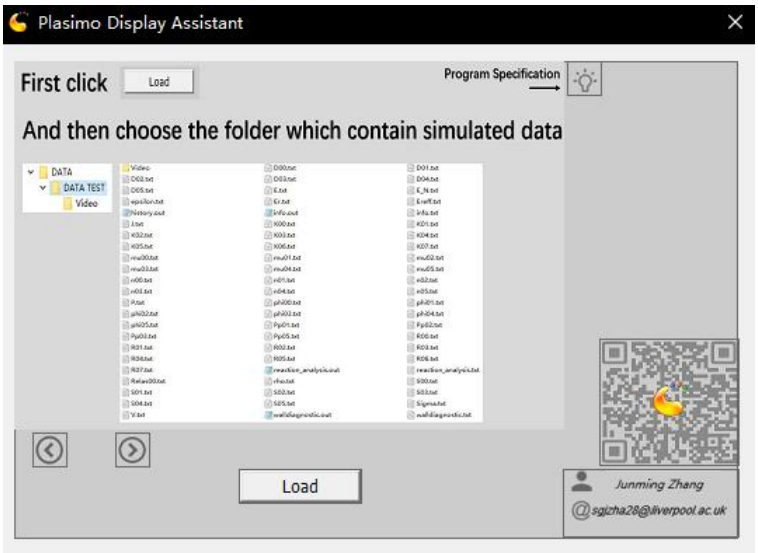


Figure 50. The second page of homepage

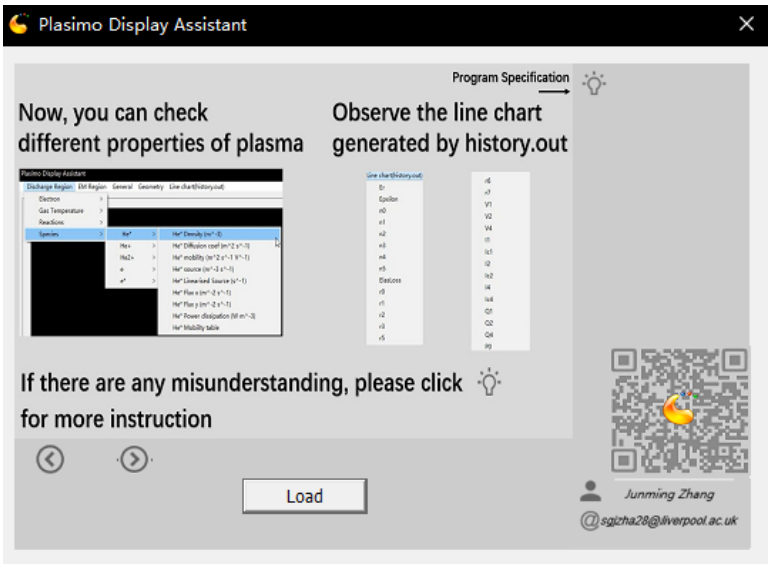


Figure 51. The third page of homepage

Load output file from Plasimo:

The output data of Plasimo is stored in one folder. This program will load all data into memory when users choose the address of this folder. Firstly, this program will display the final output result which is stored in the info.out file. This function is shown in Figure 53.

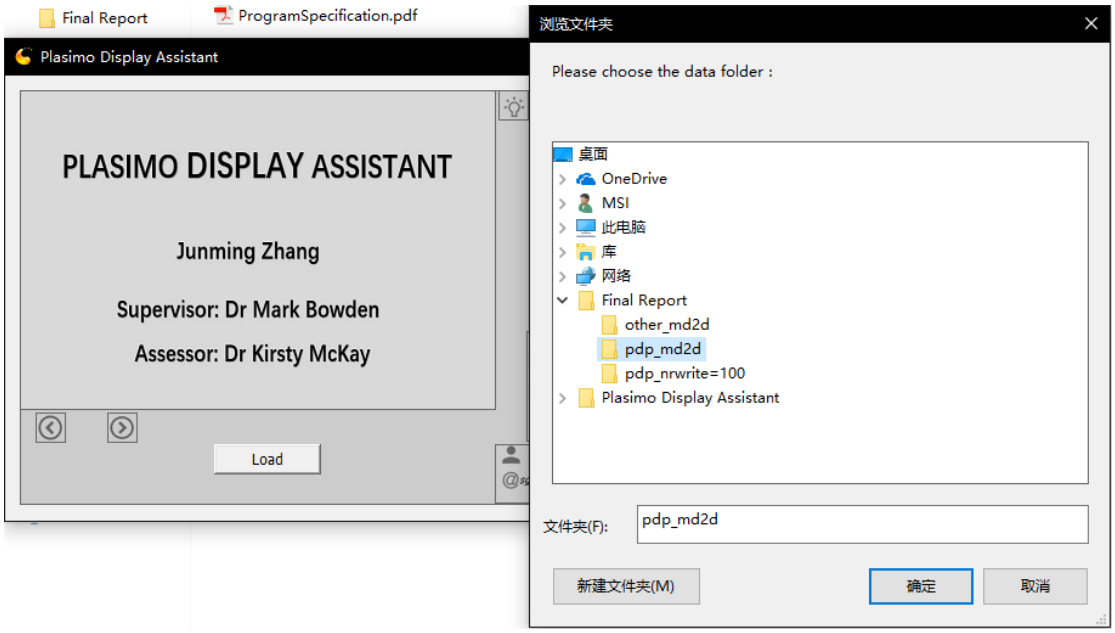


Figure 52. Load the output file into program

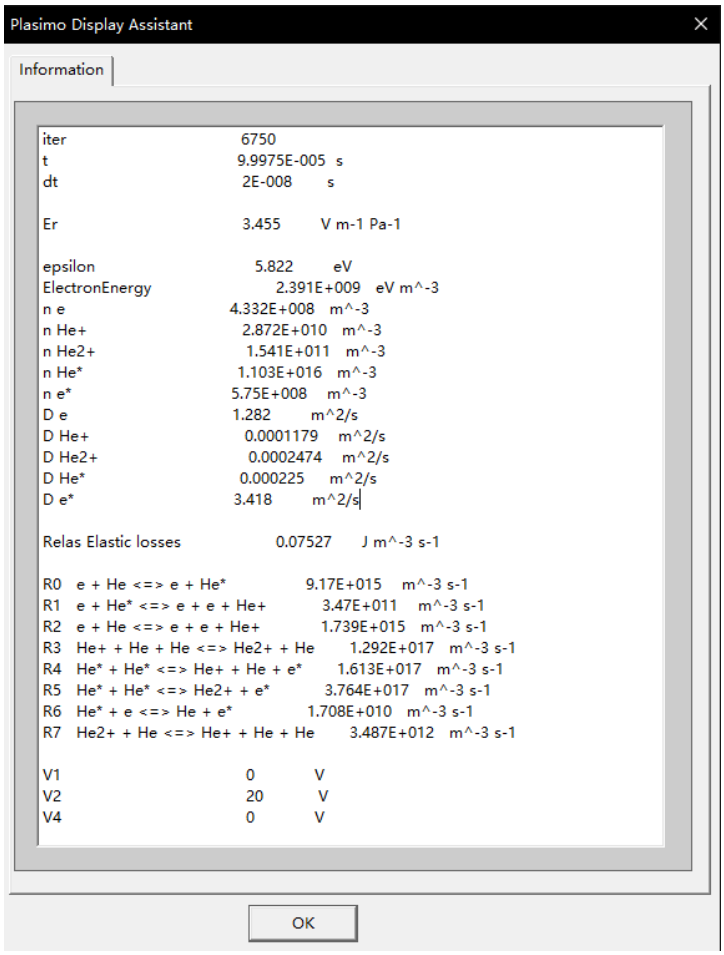


Figure 53. The final output information of input model

Main window of this project:

The main window can be divided into three parts. The top area is the menu, left area is the place to display the 2D graphical image, the right area is the place to show the numerical data.

The menu is shown in Figure 54. It different from the prototype design because pull-down menu can save the space, it makes the menu combine with the main function areas. Therefore the users can operate this program in one window. There are 68 output file and 87 properties of the pdp_md2d model.

2D graphic image and numerical data display:

The digital data will show on the right side of the main window. The 2D graphic images will be generated based on these data. Data in one period is corresponding to one image, the 2D graphic image will be divided into 7 different intervals with different colors and the digital range will be generated based on the output data. The time and each range

will put on the left side of the 2D graphic image. Users could observe the state of plasma in a particular time with numerical data at the same time. Figure 55 is the demonstration of these two function.

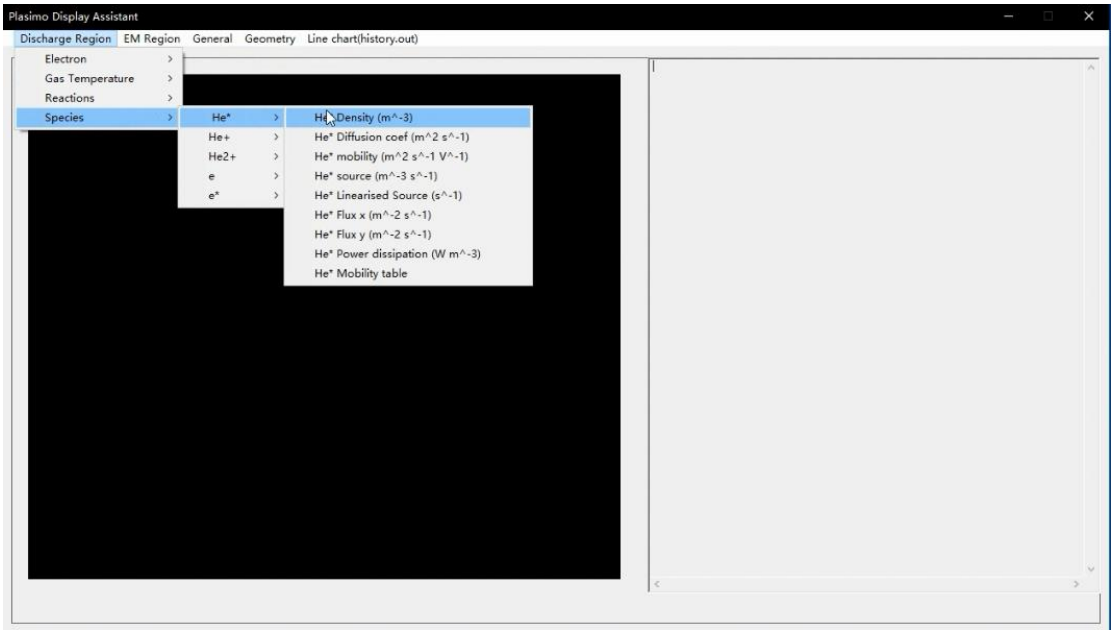


Figure 54. The main window of program and pull-down menu

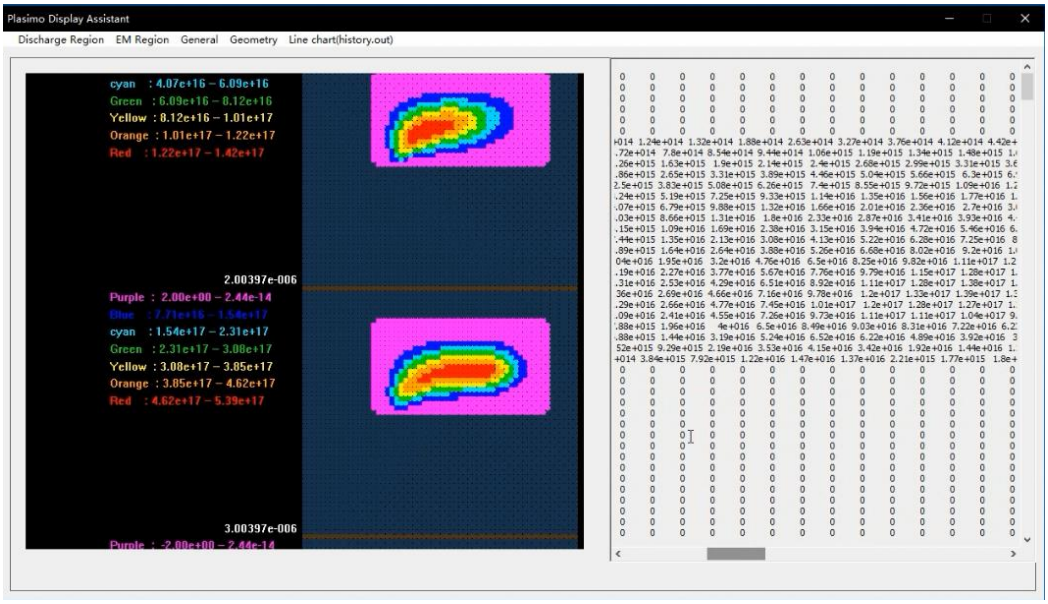


Figure 55. The 2D graphical image with digital number

Line chart based on data in history.out:

The user usually checks the data in history.out. A line chart function is used to help them observe the output data conveniently. This menu is on the rightmost side of the main menu. The time which is the first column data in history.out is used to generate the x-axis and the numerical data of other columns are used to produce the y-axis. The scale of the coordinate system will be generated automatic. The line chart function is shown in Figure 57.

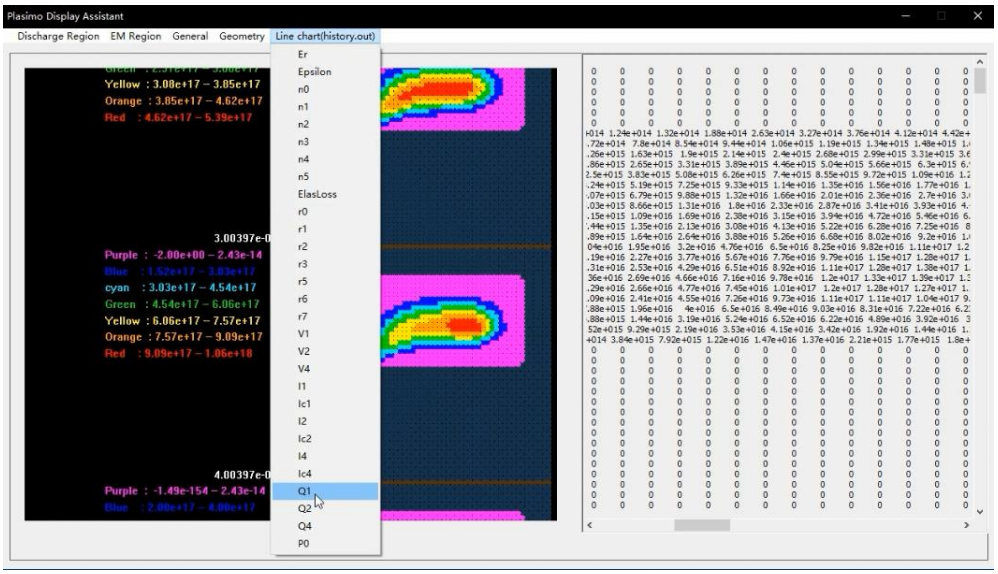


Figure 56. The line chart menu

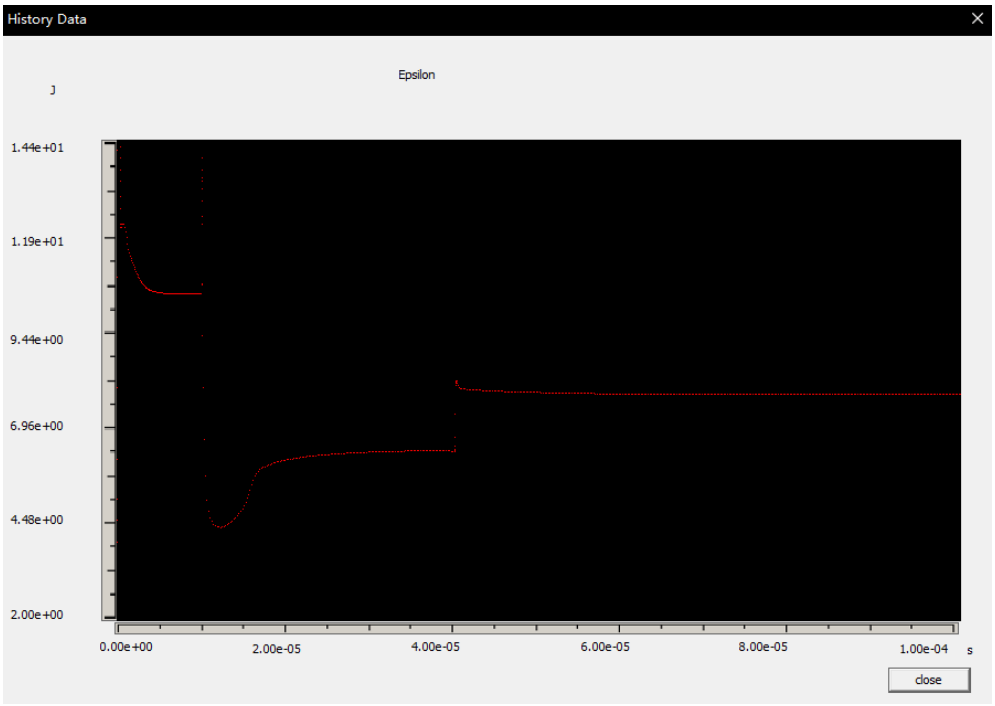


Figure 57. The line chart function

7. Industrial Relevance and Impact

There are lots of charged particles in plasma which let it has different characteristics with other typical gas:

- High electrical conductivity and electromagnetic coupling.
- It consists of different particles and strong interaction force between them.
- Surface Plasmon Polariton
-

The fundamental principle of the plasma is the interaction in charge particles under the influence of electric field and magnetic field. A variety of applications can be achieved by using these characteristics.

For example, low-temperature plasma sources and gas discharges are found with high value in atmospheric, industrial, astrophysical and domestic [9].

In order to investigate different models of plasma, the most efficient method is using computer program simulate the whole changing process and obtain related data, and then display the useful information with dynamic graph or image. According to the observation of the output data and changing process, the mistake in model or technology can be found in the simulation part. In other words, one simulation is equal to one virtual experiment. Normally real plasma experiment is very expensive. However, the simulation is fast and low cost, users could reduce the unnecessary times of design and experiment. Moreover, the optimization of the technological process could be used to improve the production efficiency.

Plasimo is one of the plasma simulated tool and it is the target in this project. It supports the completed simulation of plasma sources in numerous balanced condition.

Plasimo can be used to do a simulation of lots models of plasma.

These applications are found on the homepage of Plasimo [10]:

- Coaxial Plasma Waveguide: Microwave plasma source for large are deposition, decontamination or passivation of layers.
- Chemical Vapour Deposition: Chemical reduction techniques.
- Surfatron: Microwave surfatron plasma for optical fibre production.
- PCVD: Chemical Vapor Deposition for optical fibre production.
- Transport and radiation in complex LTE mixtures

- Sputtering hollow cathode discharge: Sputtering hollow cathode discharges for metal vapour lasers.
- Cascaded arc: Magnum-PSI plasma for testing of materials for the ITER divertor.
- Microplasmas: Microplasmas for biomedical applications.
- Metal-halide lamp: Colour segregation in the metal-halide lamp.
- ICP torch: An open atmospheric inductively coupled plasma.
- Traditional fluorescent lamp: Traditional fluorescent lamp.
- Compact fluorescent lamp: Ignition of a compact fluorescent lamp.

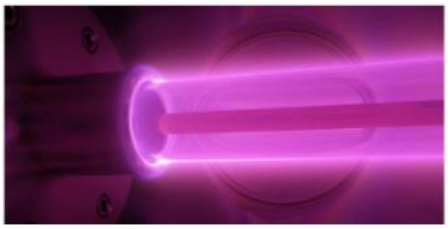


Figure 58. Coaxial Plasma Waveguide

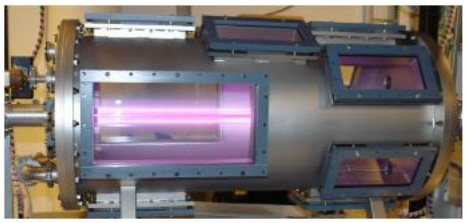


Figure 59. Chemical Vapour Deposition

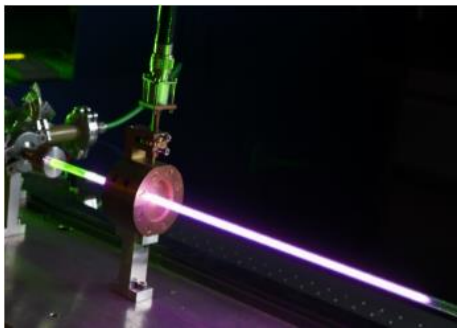


Figure 60. Surfatron

Figure 61. PCVD

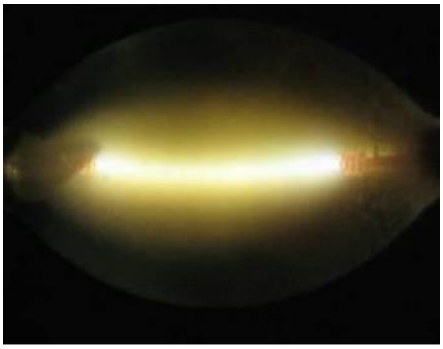


Figure 62. Transport and radiation in complex LTE mixtures

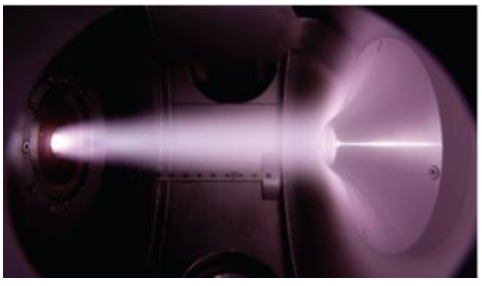


Figure 63. Cascaded arc

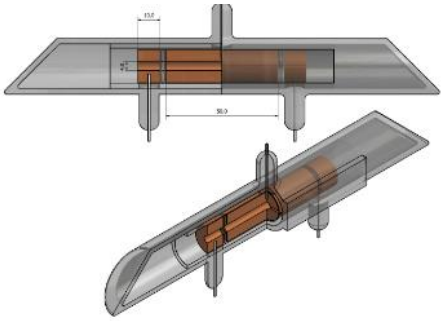


Figure 64. Sputtering hollow cathode discharge

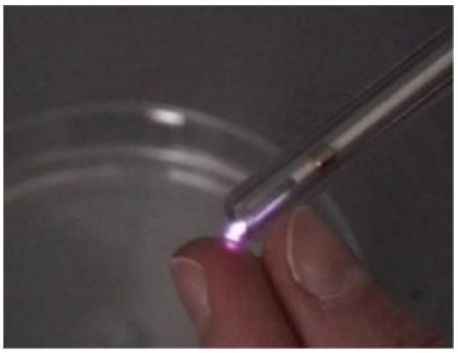


Figure 65. Microplasmas

Figure 66. Metal-halide lamp

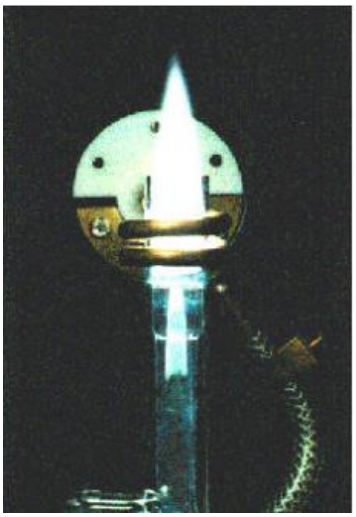


Figure 67. ICP torch

Figure 68. Traditional fluorescent lamp

Figure 69. Compact fluorescent lamp

In this project, a Plasimo assistant interface is used to process output data from the simulation tool, which means the advantage of Plasimo will be amplified:

- Figure 70 is the primary interface of this project. This program already pre-processed the output data and displayed them in 2D graphical with different colors. Therefore, users are more convenient to observe the changing process and numerical data.

- Figure 71 is the line chart function of this project. This function supports an essential function which Plasimo users usually use in the regular analysis of each common data in plasma. Therefore, it supports a different way to observe the data and give help to investigate them.

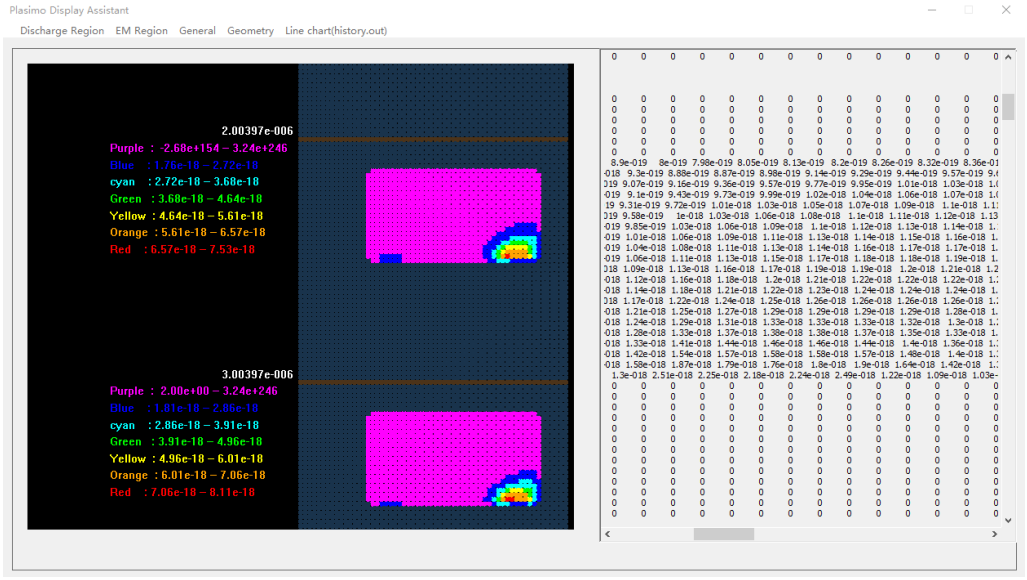


Figure 70. The operation window of Plasimo display assistant

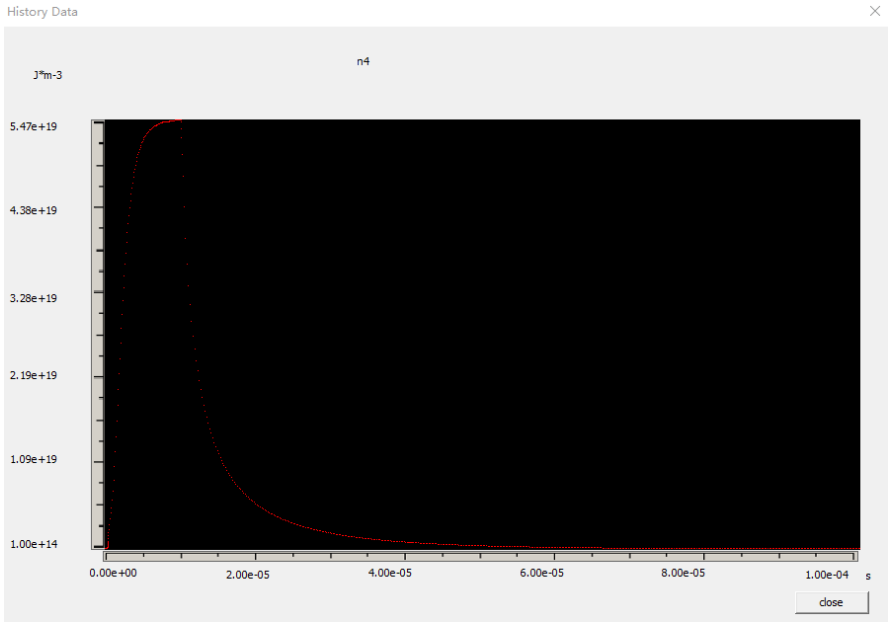


Figure 71. The line chart function of pdp_md2d model of Plasimo display assistant

Figure 72 shows the CO2 emissions situation around the word, the emissions of developing world continues increasing because they keep developing industry. At the same time, the resources depletion is a hard problem for all people. The consumable speed of fuel is more than their regeneration rate.

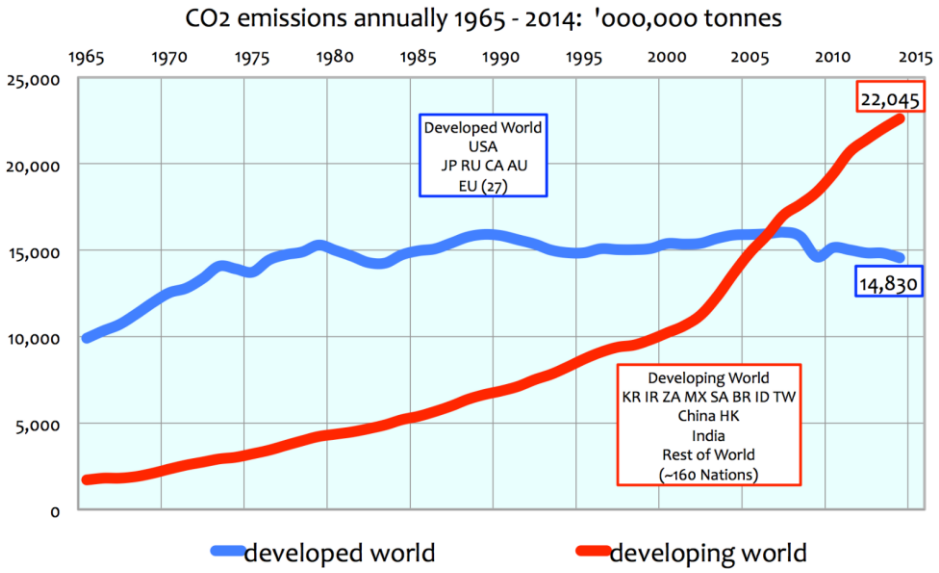


Figure 72. The CO2 emissions line chart of developed and developing countries between 1965-2014 [11]

CO2 emissions and resources depletion can be both solved at the same time by converting CO2 into fuels that can directly replace fossil fuels [12]. A current CO2 model is investigated by Plasimo. It uses the plasma source to obtain oxygen by converting CO2 into CO. However, the modeling process was difficult to finish because various molecules have lots of different states which make the modeling project complex. This technology will change the world when it comes to success. The social development will achieve a new level because of clean and cyclic energy.

8. Discussion

8.1 Interface testing

Three output data of Plasimo will be used to test this program. The number of NrWrite control the output information of the md2d model.Higher NrWrite will generate more data.

- Pdp_md2d model: NrWrite=10 and file size is 32.9MB.
- Pdp_md2d model: NrWrite=100 and file size is 243 MB.
- Other_md2d model: file size is 121MB, in this model, the 2D graphical image almost fill in space because of less 0 and 1 in the output file.

At present, this program can process 150 segments in one text file. But this number can be increased in the MyOpenGLView1.cpp file, change the number followed by the #define maximum to control the maximum number of periods in one text file. This step is shown in Figure 73.

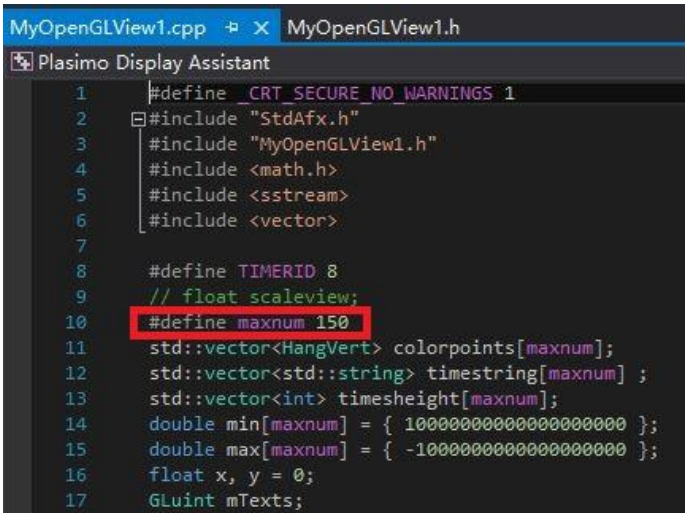


Figure 73. The maximum number of times in one text file setting

Pdp_md2d model with NrWrite 10:

It cost 1.15s to finish the 2D graphical image generation and numerical data display function. It was running smoothly in both data and line chart display function. The test result is shown in Figure 74 and 75.

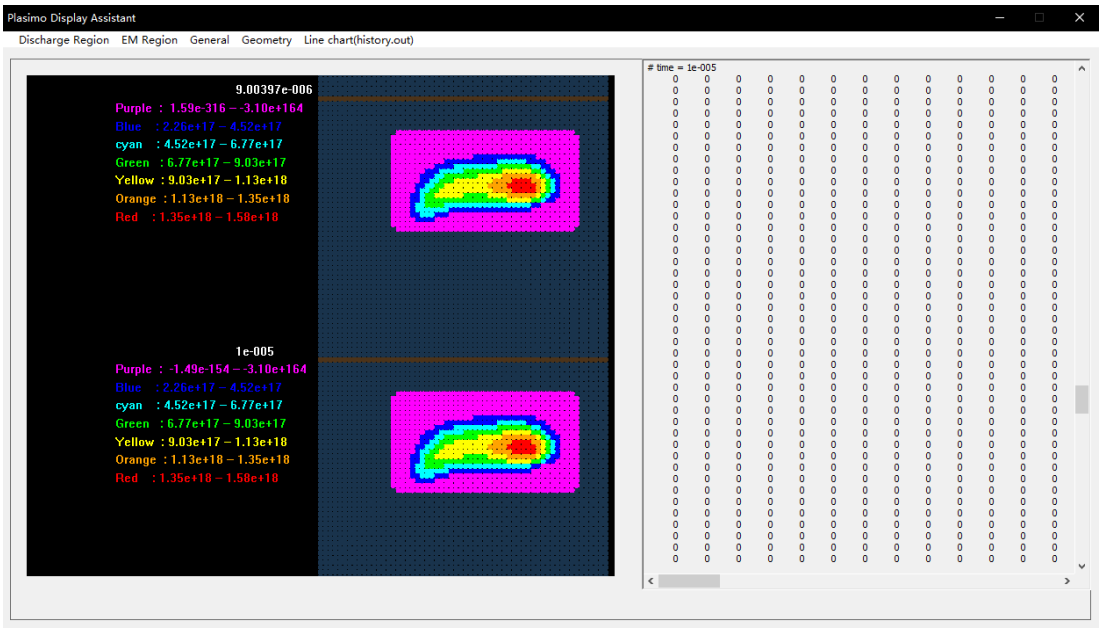


Figure 74. The test result of pdp_md2d model with NrWrite 10 in data display function

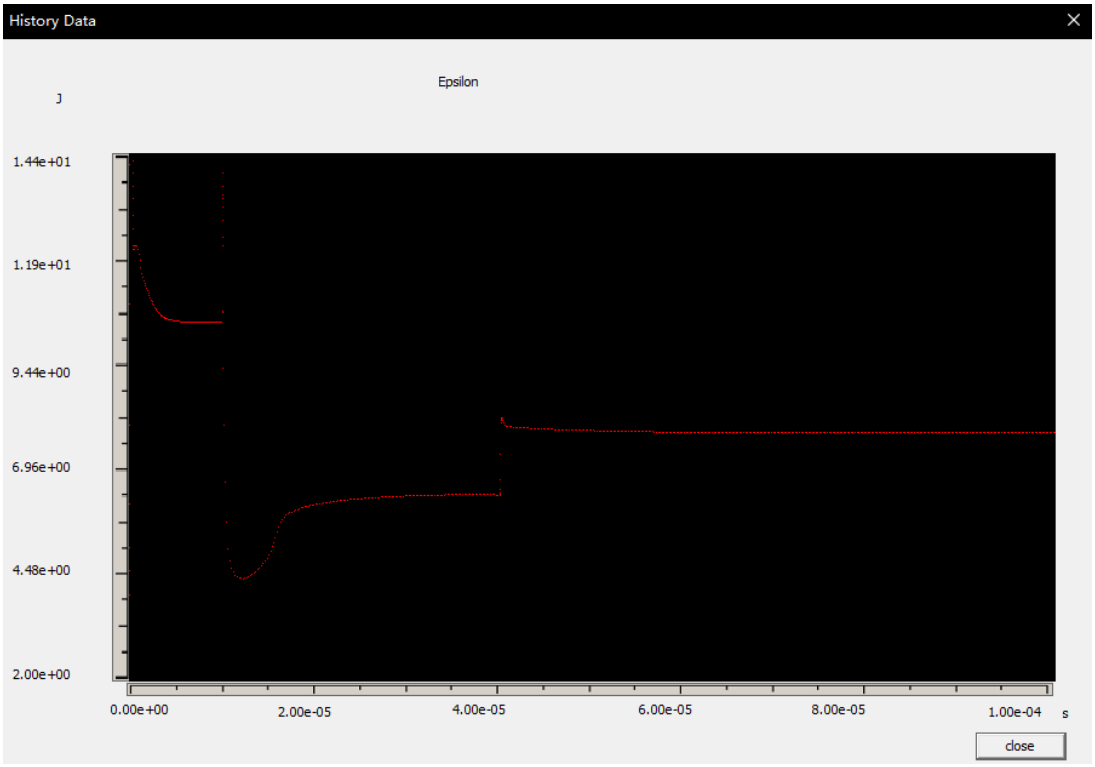


Figure 75. The test result of pdp_md2d model with NrWrite 10 in line chart display function

Pdp_md2d model with NrWrite 100:

It cost 5.23s to finish the 2D graphical image generation and numerical data display function. It was not smoothly as the previous one in the data display part because the input data are 8 times of the previous size. The line chart result is same as the previous because they are the same model. The test result is shown in Figure 76 and 77.

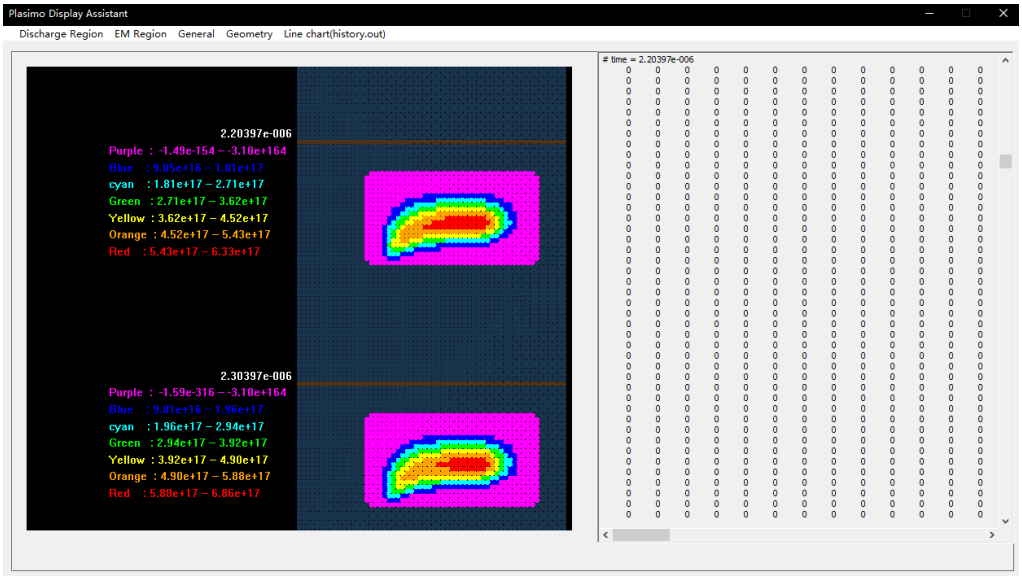


Figure 76. The test result of pdp_md2d model with NrWrite 100 in data display function

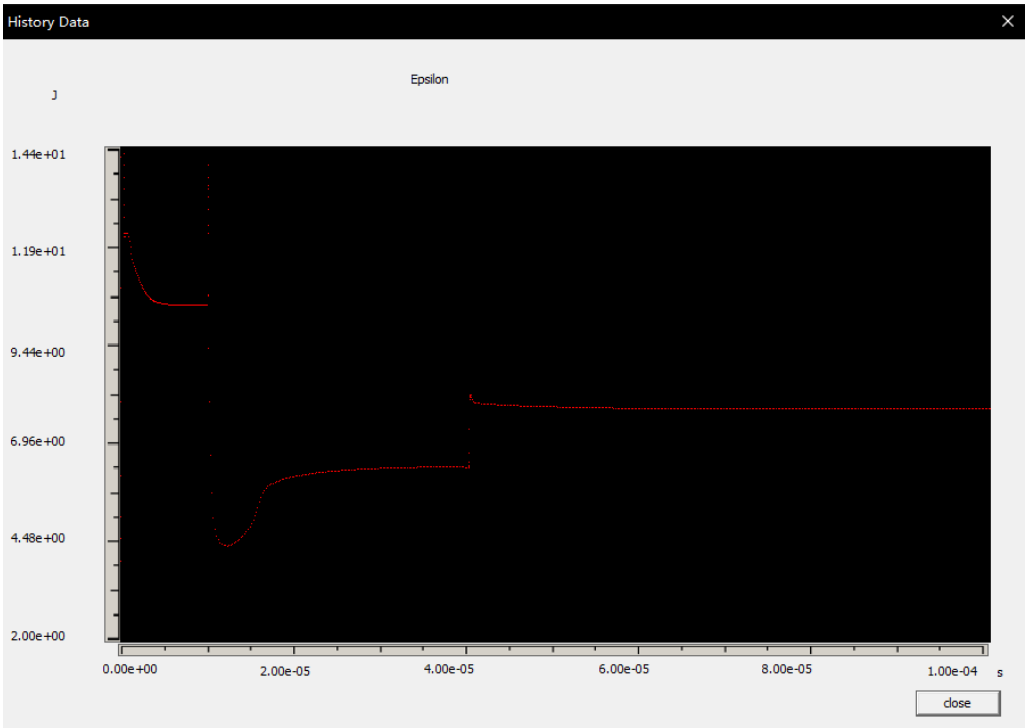


Figure 77. The test result of pdp_md2d model with NrWrite 100 in line chart display function

Other md2d model:

It cost 3.46s to finish the 2D graphical image generation and numerical data display function. It was running smoothly in both data and line chart display function. In this model, the output data are not same with the pdp_md2d file. Therefore the generated image and line chart are different from it which means this program can be applied at the related md2d model. The test result is shown in Figure 78 and 79.

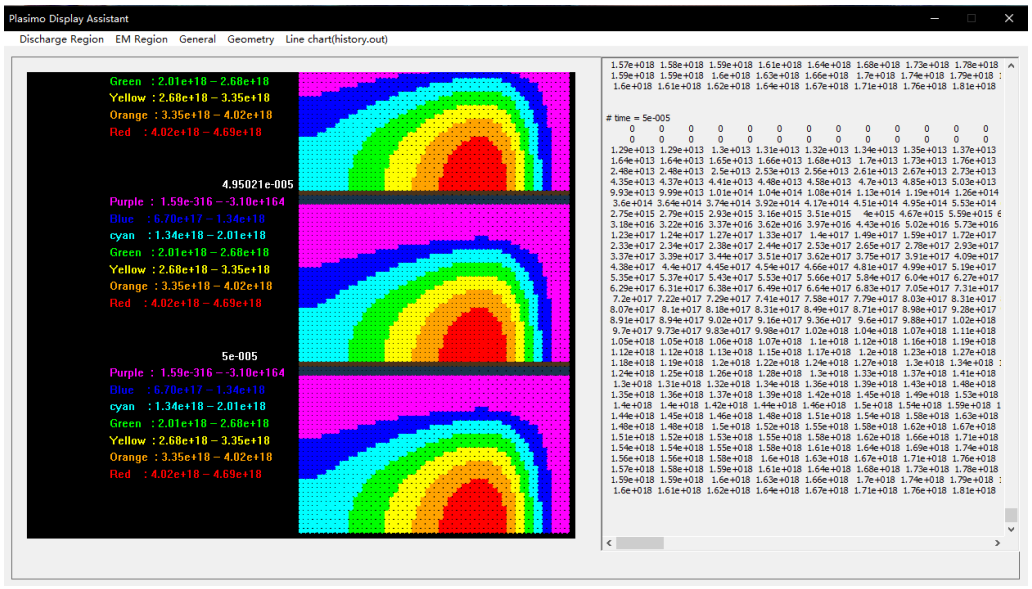


Figure 78. The test result of other md2d model in data display function

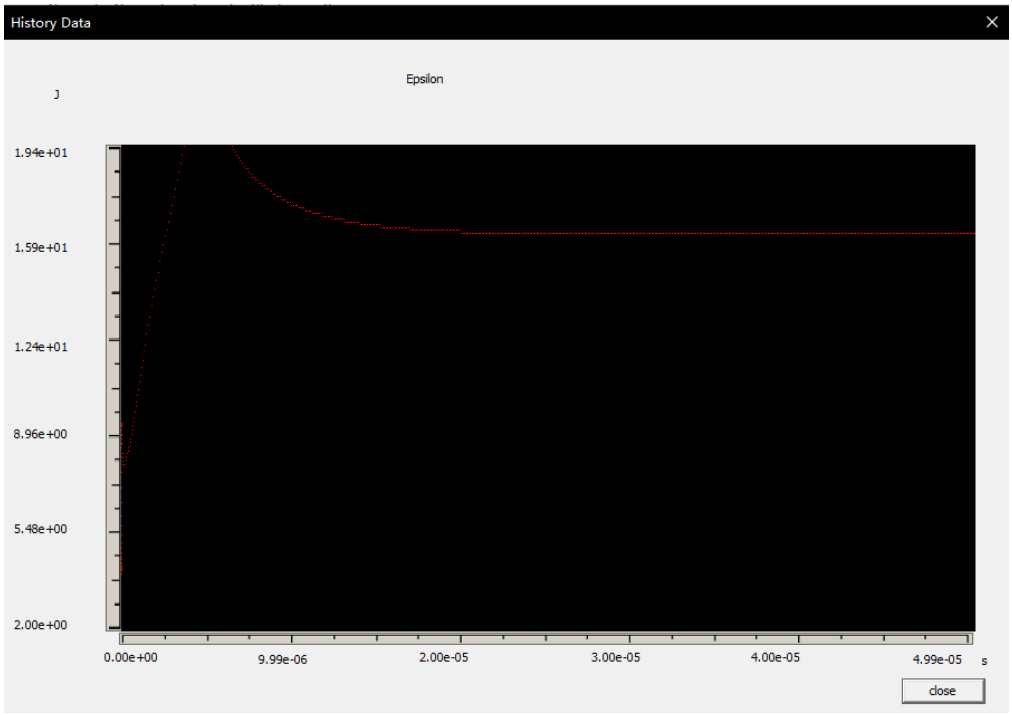


Figure 79. The test result of other md2d model in line chart display function

8.2 Project assessment

Achievements:

In Plasimo, the numerical output data are saved in 68 different text files and their name is unreadable, but the developed interface matching most text files to corresponding properties. Therefore, the users can find required data easily.

In Plasimo, the simulated process is irreversible and time-consuming, but he developed program can process the output data from Plasimo quickly. For example, Plasimo usually needs 6 minutes to finish one simulation of the pdp_md2d model. The developed program only need 1.15s to complete the data processing. Thus, users could observe different properties conveniently.

In Plasimo, users can only observe one property at one time and they cannot obtain the simulated time and numerical data in this dialog box. But the developed program display the numerical data and label the time of the 2D graphic image. Therefore, the users could obtain the information at the same time.

Three models had tested in the previous section. In general, this interface is working successfully because compare it to the original 3D graphical model in Plasimo, the generated image is almost the same.

Visual Studio is the primary developing tool and all code is written by C++ style which is meet the retirement of this project.

Drawbacks:

The processing speed will slow down with the increasing size of the input file. In the test section, it cost 5.23s to process a 243MB file. In the real-life simulation, some models will generate large data. This program may become slower in that case.

The generated 2D graphical images are controlled by the mouse wheel, if the program produced many images, users might need more time to find the image which at the specific time.

8.2 Addition exploitation

In this project, pdp_md2d is the target model. However, it is just one model of the Plasimo. According to the investigate of Plasimo, it could found that different model will generate different files with a different name. Therefore, the addition exploitation will focus on the extensional function of this project.

The matched result of files and corresponding properties will stored in a CSV file in the NameManagement folder. For example, the pdp_md2d model generates 68 output files. However, there are 87 properties of this model. Some output file may be lost because the original code of pdp_md2d model limited these files.

The CSV file can quick manage the matching result. Figure 80 display the basic theory of matching. The sub menu is marked by number 1 to 87 because there are 87 properties in pdp_md2d model, The CSV file will store the corresponding text file name based on the number 1 to 87. The CSV reading command will process theses data and use ShowFile command to display required data. ShowFile(0) is corresponding to the first number of CSV and sub menu.

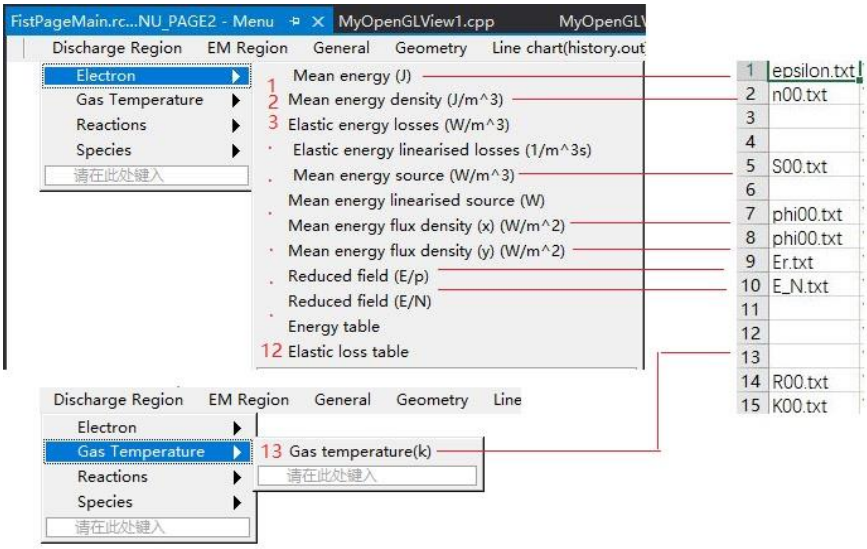


Figure 80. The basic theory of matching data and properties

```
CString ** CSVRead::ReadCsvFile(CString filepath, char divided,int *row,int *col)//Read csv file, get related data and save into an arrary

   CStringArray * Array = NULL;

   if (!PathFileExists(filepath))
       return NULL;
   CSVRead read(filepath);

   int Rowcount = read.FileRowCount();
   Array = read.CSVInf(divided);
   int Columcount = Array->GetSize();

   CString **p = new CString *[Rowcount];
   for (int i = 0; i < Rowcount; i++)
       p[i] = new CString[Columcount];
   for (int i = 0; i < Columcount; i++)
       p[0][i] = Array->GetAt(i);

   *row = Rowcount;
   *col = Columcount;
   for (int i = 1; i<Rowcount ; i++)
   {
       Array = read.CSVInf(divided);
       for (int j = 0; j < Columcount; j++)
       {
           CString a = Array->GetAt(j);
           p[i][j] = Array->GetAt(j);
       }
   }

   return p;
```

Figure 81. The code of reading csv file function

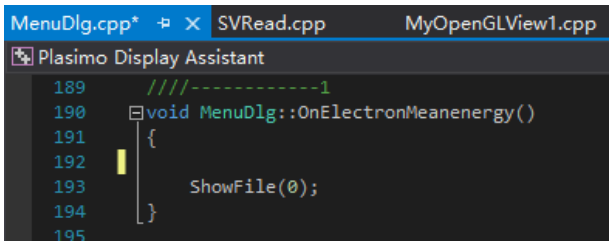


Figure 82. Display corresponding data when click sub-menu

8.3 Difficulties

This project has an abandoned version which is using videos and images to display the output data from Plasimo. This interface is shown in Figure 83. It's inapplicable because it needs lots of time to record videos. For example, it took 8 hours to record 87 videos and images of the pdp_md2d model.

It means two interfaces had been developed in 20 weeks. Therefore the time limitation is one thorny problem. Cost more time per one week is the solution to this issue.

The unfamiliar object is another dilemma. Plasma is a new area of Electrical and Electronic Engineering undergraduate. To understand the real requirements of plasma researcher, it needs more time to think. In addition, C++ and OpenGL code is almost a new area because although the fundamental knowledge of C++ was learned in Year 1, but this project need more knowledge to achieve the objective.

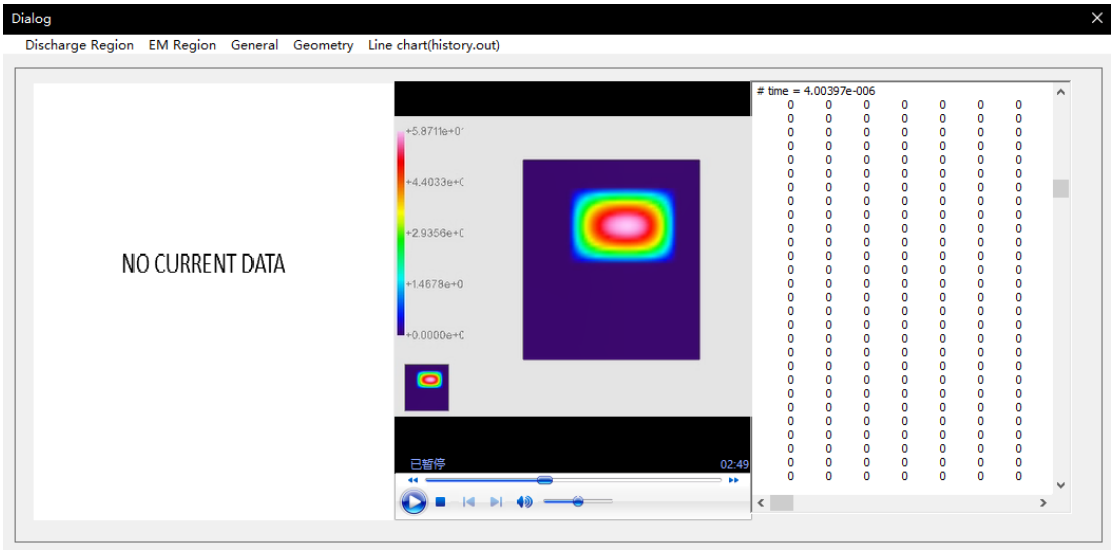


Figure 83. The video version of project

8.4 Future Improvements

In the prototype design, there was a quick menu. Users could add frequently-used properties in this menu because some features are useless to research. However, due to the time limitation, this function cannot realize.

As mentioned in the project assessment section, the processing speed of this program will slow down with the increasing size of the input file and the generated 2D graphical images are controlled by the mouse wheel.

Therefore, the future improvement is redevelopment the program, add the quick menu function and the 2D graphical image located function based on specific time. At the same time, optimizing the processing speed of input data.

9. Conclusion

In conclusion, this report focus on the software development and the objective of this project is to help Plasimo researchers obtain the simulated result more efficient by using a C++-based user interface. This report gives the general designing and developing procedures of a program. After that, the industrial relevance was mentioned to analysis the reliability of this interface and finally give some result of the design. As shown in the result, this project is achieved the necessary requirements in the analysis part. An interface was developed and it can be used to pre-processing the output data from Plasimo. In addition, it could generate the 2D graphical image based on the input data and create line chart based on specific data. Therefore, this project is successful. However, the quick menu and further program majorization can make this interface better. The knowledge of prototype design, software development by Visual Studio has learned through this project.

References List

[1] The Plasimo Team. IEEE Citation Reference [online]. Available: https://plasimo.phys.tue.nl/plasimo_overview.html (accessed 28th April 2017)

[2] The Plasimo Team. IEEE Citation Reference [online]. Available: https://plasimo.phys.tue.nl/physics/md2d/index.html (accessed 28th April 2017)

[3] The Plasimo Team. (2014 April) IEEE Citation Reference [online]. Available: https://plasimo.phys.tue.nl/generated-docs/plasimo-5.0.0/misc-docs/user_guide.pdf (accessed 28th April 2017)

[4] Junming, Zhang, Final Year Project Resubmission of Preliminary Report, University of Liverpool, November 2016

[5] Junming, Zhang, Final Year Project Resubmission of Preliminary Report, University of Liverpool, November 2016

[6] B. Shneiderman. (2010) IEEE Citation Reference [online]. Available: https://www.cs.umd.edu/users/ben/goldenrules.html (accessed 28th April 2017)

[7] Junming, Zhang, Final Year Project Resubmission of Preliminary Report, University of Liverpool, November 2016

[8] Junming, Zhang, Final Year Project Resubmission of Preliminary Report, University of Liverpool, November 2016

[9] V. Jan, M. Anna and Y. Vasil. (2009) IEEE Citation Reference [online]. Available: http://aip.scitation.org.liverpool.idm.oclc.org/doi/pdf/10.1063/1.3141691 (accessed 28th April 2017)

[10] The Plasimo Team. IEEE Citation Reference [online]. Available: https://plasimo.phys.tue.nl/applications/index.html (accessed 28th April 2017)

[11] M. Euan. (2016) IEEE Citation Reference [online]. Available: http://euanmearns.com/the-record-of-recent-man-made-co2-emissions-1965-2014/ (accessed 28th April 2017)

[12] The Plasimo Team. IEEE Citation Reference [online]. Available: https://plasimo.phys.tue.nl/applications/globalmodels/index.html (accessed 28th April 2017)

Appendices

Appendix 1. Original specification report form

UNIVERSITY OF LIVERPOOL

DEPARTMENT OF ELECTRICAL ENGINEERING AND ELECTRONICS

Project Specification Form 2016-2017
Final Year BEng (ELEC340) and Year 3 MEng (ELEC440)

Student Name: Junming Zhang Module: ELEC340

Supervisor: Mark Bowden Student ID No: 201138928

Project Title: Development of a C++ based user-interface for a plasma simulation tool

Project Specification

A. Project Description and Methodology:
(Overall view of the project with proposed route to realization i.e. what are the project aims and objectives and how you are going to do it?)

This project develops a practical C++ based user-interface to help researchers and industry users obtain the output of plasma simulation effectively.

The existing simulation tool of plasma will generate large amounts of output data in various formats. Thus, the main objectives of this project contain research of plasma to select numerous output data and deep learning of C++developing tool to build the user-interface. Specifically, it requires leaning of C++ based tools which contain Visual Studio (C++ fundamental), Qt (develop GUI) and OpenGL (build 3D graphics).

In order to achieve them within 20 weeks, background reading and research for plasma discharges should be the first objective, at the meanwhile, the skill of C++ developing tools should be practiced. Next, the developing of this software should be proceeded. Finally, the report and presentation of this project should be prepared.

B. Project Tasks and Milestones: (indicate the tasks and milestones that should be achieved and their expected dates e.g. understanding of theory, designs of circuits, construction of circuits, software specifications, working demonstrations etc.)

Tasks: (a task is a package of work that should be completed during a particular time period)

Preparatory Work Week 1 ~ Week 2
- Obtaining plasma simulation tool and relevant reading materials.

Research Work Week 3 ~ Week 15
-Weekly background reading for plasma discharges.
- Investigating and selecting significant data in numerous output files from plasma simulation tool.
- Investigating and comparing different interface of learning, researching and factory software.

Developing work Week 4 ~ Week 15
- Learning of corresponding C^{++} based software include openGL, visual studio and Qt.

- 3D graphical model display function developing.
- Write relevant report.

- Rapid loading text function developing.
- Write relevant report.

- Intuitive user-interface developing and combine all functions together.
- Write relevant report

-Software testing and optimization.

Report working Week 1 ~ Week 20
- Project specification report form writing.
- Preliminary report writing.
- Preparation for first presentation
- Creating poster.
- Preparation for bench inspection.
- Final report writing.
- Weekly virtual log book

Milestones: (an objective that should be achieved by a particular date e.g. the completion of a task)
Semester 1
Week 3 Completion of preparatory work, project specification and preliminary report writing. **Week 11** Completion of first presentation preparation. Completion of rapid loading text function and 3D graphical model display function developing as necessary elements of user-interface.
Semester 2
Week 15 Completion of user-interface developing and whole system testing. Completion of virtual log book. **Week 20** Completion of poster, bench inspection presentation and final report writing.

C. Project Deliverables: (Indicate what should be completed at the end of the project e.g. this list should indicate what will be presented / demonstrated at the final bench inspections)

- The completed software of plasma simulation user-interface with detailed software handbook.

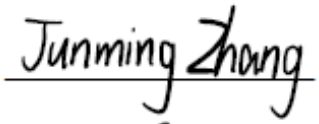
- The poster with significant achievements of this project.

- The software demonstration and presentation.

- The main functions will be displayed and introduced with the software and codes.

- The final report and relevant documents of this project.

D. A section on Project Rationale and Industrial Relevance must be included in the preliminary report (deadline midnight Friday 14th October 2016). This should explain how and why the project was devised, e.g. it may be a project sponsored by a company or linked to a research project.

Student Signature:  Date: 11/10/2016

Supervisor's Signature:  Date: 11/10/2016

By signing this form, the supervisor and student are confirming that the project is of a sufficiently demanding nature that it is suitable for the individual project component of an accredited engineering degree and that a student, who is capable of producing a first class performance, will be able to demonstrate his/her capabilities in this project.

Appendix 2. Revised specification report form

UNIVERSITY OF LIVERPOOL

DEPARTMENT OF ELECTRICAL ENGINEERING AND ELECTRONICS

Revisionary Project Specification Form 2016-2017
Final Year BEng (ELEC340) and Year 3 MEng (ELEC440)

Student Name: Junming Zhang Module: ELEC340

Supervisor: Mark Bowden Student ID No: 201138928

Project Title: Development of a C++ based user-interface for a plasma simulation tool

Project Specification

A. Project Description and Methodology:

This project develops a practical C++ based user-interface to help researchers and industry users analyze the output of plasma simulation effectively.

The existing simulation tool Plasimo will generate large amounts of output data in various formats. Thus, the main objectives of this project contain design prototype of interface and deep learning of C++ to developing this program. Specifically, it requires me to learn C++ based tools which contain Visual Studio (build GUI) and OpenGL (build 2D graphics).

In order to achieve them within 20 weeks, I need to do user requirements research first, at the meanwhile, the skill of developing C++ need to train. Next, the developing of this software need to proceed. Finally, the report and presentation of this project need to prepare.

B. Project Tasks and Milestones:

Tasks: (a task is a package of work that should be completed during a particular time period)

Preparatory Work Week 1 ~ Week 2
- Obtaining plasma simulation tool and relevant reading materials.

Research Work Week 2 ~ Week 15
-Weekly background reading for plasma discharges.
- Investigating significant data in numerous output file.
- Investigating Interface in different researching and factory software.
- Learning of corresponding software include C^{++} based visual studio, OpenGL.

Developing work Week 4 ~ Week 15
- 2D graphical model display function developed by OpenGL.
- Rapid loading text function developed by Visual Studio.
- User-friendly interface developed by Visual Studio.

Report working Week 1 ~ Week 20
- Project specification report form writing.
- Preliminary report writing.
- Weekly virtual log book
- Preparing presentation.
- Developing poster.
-Final report writing.

Milestones: (an objective that should be achieved by a particular date e.g. the completion of a task)

Week 3
- Obtaining plasma simulation tool and relevant reading materials.
- Investigating how to build an user-friendly interface of learning.
- Researching relevant software.
- Writing project specification report form.
- Writing preliminary report.
- updating virtual log book.

Week 7
- Learning C++ and OpenGL.
- Interface prototype design.
- Developing the main page of the program.
- Update virtual log book.

Week 9
- Developing the reading function to obtain data in text files.
- Creating the content of plasma properties.
- Taking videos of plasma properties.
- Matching videos to corresponding properties.
- Update virtual log book.

Week 11
- Developing the line chart function based on input data.
- Writing presentation speech draft.
- Creating presentation PPT.
- Update virtual log book.

Week 13
- Program testing.
- Interface optimization.

Week 17
- Optimization of line chart function
- Developing the 2D graphical function
- Final debug and testing.

Week 20
- Creating poster
- Preparing presentation at the bench.

C. Project Deliverables:

- The completed software of plasma simulation user interface with detailed software handbook.

- The poster with significant achievements of this project.

- The software demonstration and presentation.

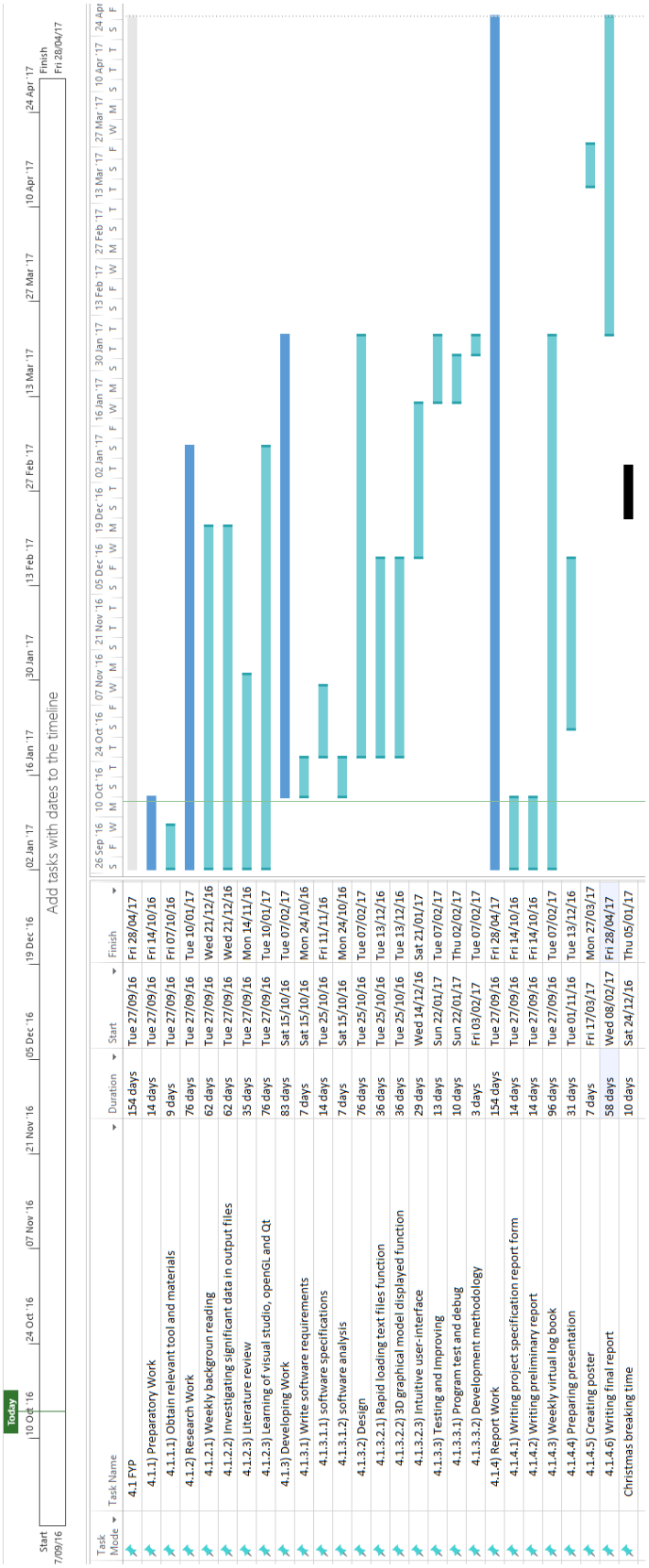
- The main functions will be displayed and introduced with the software and codes.

- The final report and relevant documents of this project.

D. Project Rationale and Industrial Relevance:

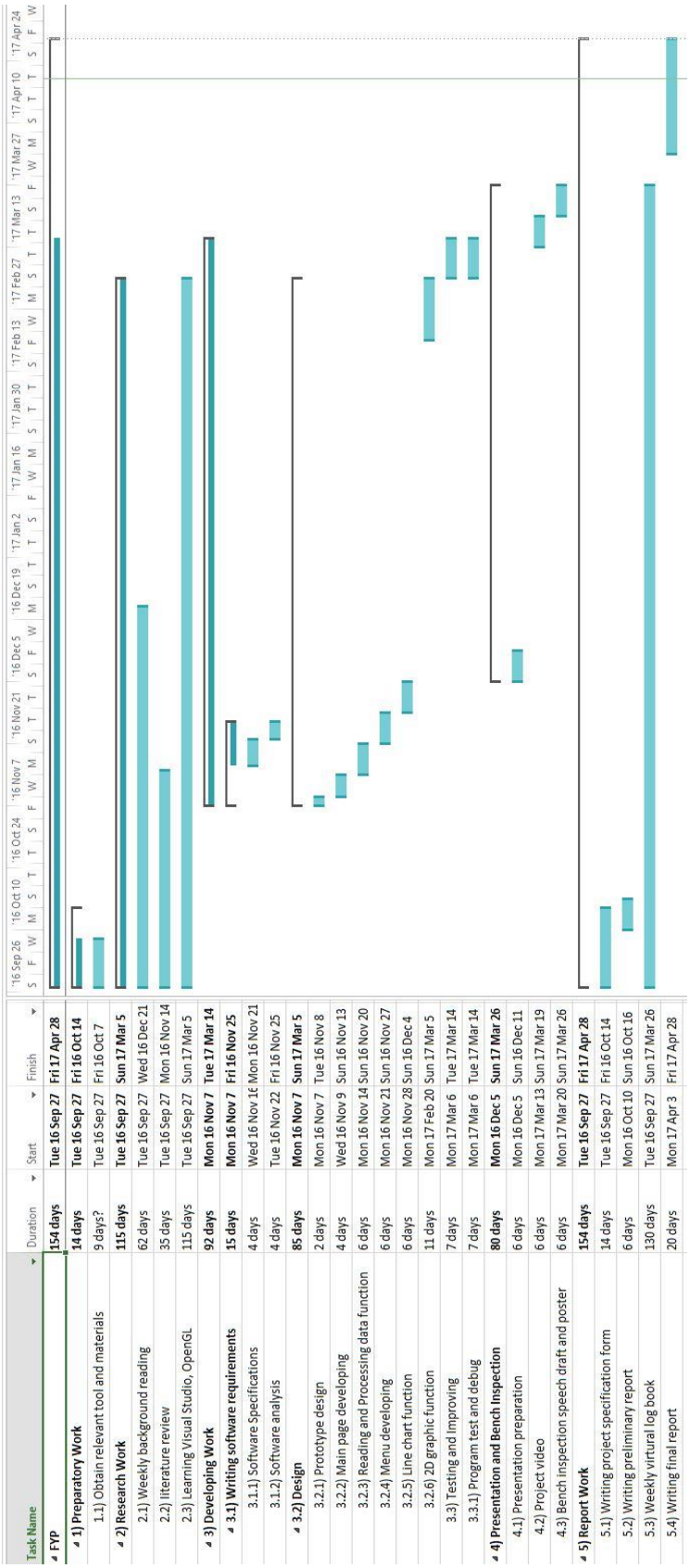
In this project, the main areas are computer science and physical, which are both significant branches of human scientific and technological development. In addition, the relationship between these two areas and EEE are not closely. Therefore, this is a challenging opportunity to test the learning ability of new areas as an EEE undergraduate.

Appendix 3. Original Gantt chart preferable produced by MS Project



Task Mode	Task Name	Duration	Start	Finish
	4.1 FYP	154 days	Tue 27/09/16	Fri 28/04/17
	4.1.1) Preparatory Work	14 days	Tue 27/09/16	Fri 14/10/16
	4.1.1.1) Obtain relevant tool and materials	9 days	Tue 27/09/16	Fri 07/10/16
	4.1.2) Research Work	76 days	Tue 27/09/16	Tue 10/01/17
	4.1.2.1) Weekly backgroun reading	62 days	Tue 27/09/16	Wed 21/12/16
	4.1.2.2) Investigating significant data in output files	62 days	Tue 27/09/16	Wed 21/12/16
	4.1.2.3) Literature review	35 days	Tue 27/09/16	Mon 14/11/16
	4.1.2.3) Learning of visual studio, openGL and Qt	76 days	Tue 27/09/16	Tue 10/01/17
	4.1.3) Developing Work	83 days	Sat 15/10/16	Tue 07/02/17
	4.1.3.1) Write software requirements	7 days	Sat 15/10/16	Mon 24/10/16
	4.1.3.1.1) software specifications	14 days	Tue 25/10/16	Fri 11/11/16
	4.1.3.1.2) software analysis	7 days	Sat 15/10/16	Mon 24/10/16
	4.1.3.2) Design	76 days	Tue 25/10/16	Tue 07/02/17
	4.1.3.2.1) Rapid loading text files function	36 days	Tue 25/10/16	Tue 13/12/16
	4.1.3.2.2) 3D graphical model displayed function	36 days	Tue 25/10/16	Tue 13/12/16
	4.1.3.2.3) Intuitive user-interface	29 days	Wed 14/12/16	Sat 21/01/17
	4.1.3.3) Testing and Improving	13 days	Sun 22/01/17	Tue 07/02/17
	4.1.3.3.1) Program test and debug	10 days	Sun 22/01/17	Thu 02/02/17
	4.1.3.3.2) Development methodology	3 days	Fri 03/02/17	Tue 07/02/17
	4.1.4) Report Work	154 days	Tue 27/09/16	Fri 28/04/17
	4.1.4.1) Writing project specification report form	14 days	Tue 27/09/16	Fri 14/10/16
	4.1.4.2) Writing preliminary report	14 days	Tue 27/09/16	Fri 14/10/16
	4.1.4.3) Weekly virtual log book	96 days	Tue 27/09/16	Tue 07/02/17
	4.1.4.4) Preparing presentation	31 days	Tue 01/11/16	Tue 13/12/16
	4.1.4.5) Creating poster	7 days	Fri 17/03/17	Mon 27/03/17
	4.1.4.6) Writing final report	58 days	Wed 08/02/17	Fri 28/04/17
	Christmas breaking time	10 days	Sat 24/12/16	Thu 05/01/17

Appendix 4. Revised Gantt chart preferable produced by MS Project



Task Name	Duration	Start	Finish
FYP	154 days	Tue 16 Sep 27	Fri 17 Apr 28
1) Preparatory Work	14 days	Tue 16 Sep 27	Fri 16 Oct 14
1.1) Obtain relevant tool and materials	9 days?	Tue 16 Sep 27	Fri 16 Oct 7
2) Research Work	115 days	Tue 16 Sep 27	Sun 17 Mar 5
2.1) Weekly background reading	62 days	Tue 16 Sep 27	Wed 16 Dec 21
2.2) literature review	35 days	Tue 16 Sep 27	Mon 16 Nov 14
2.3) Learning Visual Studio, OpenGL	115 days	Tue 16 Sep 27	Sun 17 Mar 5
3) Developing Work	92 days	Mon 16 Nov 7	Tue 17 Mar 14
3.1) Writing software requirements	15 days	Mon 16 Nov 7	Fri 16 Nov 25
3.1.1) Software Specifications	4 days	Wed 16 Nov 16	Mon 16 Nov 21
3.1.2) Software analysis	4 days	Tue 16 Nov 22	Fri 16 Nov 25
3.2) Design	85 days	Mon 16 Nov 7	Sun 17 Mar 5
3.2.1) Prototype design	2 days	Mon 16 Nov 7	Tue 16 Nov 8
3.2.2) Main page developing	4 days	Wed 16 Nov 9	Sun 16 Nov 13
3.2.3) Reading and Processing data function	6 days	Mon 16 Nov 14	Sun 16 Nov 20
3.2.4) Menu developing	6 days	Mon 16 Nov 21	Sun 16 Nov 27
3.2.5) Line chart function	6 days	Mon 16 Nov 28	Sun 16 Dec 4
3.2.6) 2D graphic function	11 days	Mon 17 Feb 20	Sun 17 Mar 5
3.3) Testing and Improving	7 days	Mon 17 Mar 6	Tue 17 Mar 14
3.3.1) Program test and debug	7 days	Mon 17 Mar 6	Tue 17 Mar 14
4) Presentation and Bench Inspection	80 days	Mon 16 Dec 5	Sun 17 Mar 26
4.1) Presentation preparation	6 days	Mon 16 Dec 5	Sun 16 Dec 11
4.2) Project video	6 days	Mon 17 Mar 13	Sun 17 Mar 19
4.3) Bench inspection speech draft and poster	6 days	Mon 17 Mar 20	Sun 17 Mar 26
5) Report Work	154 days	Tue 16 Sep 27	Fri 17 Apr 28
5.1) Writing project specification form	14 days	Tue 16 Sep 27	Fri 16 Oct 14
5.2) Writing preliminary report	6 days	Mon 16 Oct 10	Sun 16 Oct 16
5.3) Weekly virtural log book	130 days	Tue 16 Sep 27	Sun 17 Mar 26
5.4) Writing final report	20 days	Mon 17 Apr 3	Fri 17 Apr 28

Appendix 5. The Program Code of Project

The code is listed as follows and available on drop box.

Download link:
https://www.dropbox.com/s/hdk3be5rnm4gubx/Plasimo%20Display%20Assistant_Junming_Zhang.rar?dl=0

Head file:

CvvImage.h

```
#pragma once
#ifndef CVVIMAGE_CLASS_DEF
#define CVVIMAGE_CLASS_DEF
#include "opencv.hpp"
/* CvvImage class definition */
class  CvvImage
{
public:
    CvvImage();
    virtual ~CvvImage();
    /* Create image (BGR or grayscale) */
    virtual bool  Create( int width, int height, int bits_per_pixel, int
image_origin = 0 );
    /* Load image from specified file */
    virtual bool  Load( const char* filename, int desired_color = 1 );
    /* Load rectangle from the file */
    virtual bool  LoadRect( const char* filename,
        int desired_color, CvRect r );
#if defined WIN32 || defined _WIN32
    virtual bool  LoadRect( const char* filename,
        int desired_color, RECT r )
    {
        return LoadRect( filename, desired_color,
            cvRect( r.left, r.top, r.right - r.left, r.bottom - r.top ));
    }
#endif
    /* Save entire image to specified file. */
    virtual bool  Save( const char* filename );
    /* Get copy of input image ROI */
    virtual void  CopyOf( CvvImage& image, int desired_color = -1 );
    virtual void  CopyOf( IplImage* img, int desired_color = -1 );
```

```
    IplImage* GetImage() { return m_img; };
    virtual void  Destroy(void);
    /* width and height of ROI */
    int Width() { return !m_img ? 0 : !m_img->roi ? m_img->width :
m_img->roi->width; };
    int Height() { return !m_img ? 0 : !m_img->roi ? m_img->height :
m_img->roi->height;};
    int Bpp() { return m_img ? (m_img->depth & 255)*m_img->nChannels : 0; };
    virtual void  Fill( int color );
    /* draw to highgui window */
    virtual void  Show( const char* window );

#if defined WIN32 || defined _WIN32
    /* draw part of image to the specified DC */
    virtual void  Show( HDC dc, int x, int y, int width, int height,
        int from_x = 0, int from_y = 0 );
    /* draw the current image ROI to the specified rectangle of the destination
DC */
    virtual void  DrawToHDC( HDC hDCDst, RECT* pDstRect );
#endif
protected:
    IplImage*  m_img;
};
typedef CvvImage CImage;
#endif
```

DialogGl.h

```
#pragma once

#include <opencv2/opencv.hpp>
#include"opencv/highgui.h"
#include"CvvImage.h"
#include"resource.h"
#include"MyOpenGLView.h"


class DialogGl : public CDialog
{
    DECLARE_DYNAMIC(DialogGl)

public:
    DialogGl(CWnd* pParent = NULL);
```

```
    virtual ~DialogGl();

// data of dialog
#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_DIALOG_GL };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    DECLARE_MESSAGE_MAP()
public:
    void ShowLine(int biaohao);//Display the line chart
    void ReadCsvFile(CString pathfile, char divided);
    CMyOpenGLView *m_diaplay;
    virtual BOOL OnInitDialog();
    void DrawPicToHDC(char * pathfile, UINT ID);//load picture to widget
    CString kedu_x1;
    CString kedu_x2;
    CString kedu_x3;
    CString kedu_x4;
    CString kedu_x5;
    CString kedu_y1;
    CString kedu_y2;
    CString kedu_y3;
    CString kedu_y4;
    CString kedu_y5;
    afx_msg void OnBnClickedOk();
    afx_msg void OnBnClickedButton1();
    afx_msg void OnPaint();

    float kedu[4];
    afx_msg void OnStnClickedStaticx5();
    afx_msg void OnStnClickedStaticy5();
    CString kedu_x6;
    CString kedu_y6;
    afx_msg void OnStnClickedStaticy6();
    CString m_glName;
    CString m_name;
    afx_msg void OnStnClickedStaticGl();
};
```

FistPageMain.h

```
#pragma once

#ifndef __AFXWIN_H__
    #error "在包含此文件之前包含“stdafx.h”以生成 PCH 文件"
#endif

#include "resource.h"


// CFistPageMainApp:

//

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

public:
    virtual BOOL InitInstance();


    DECLARE_MESSAGE_MAP()
};

extern CFistPageMainApp theApp;
```

FistPageMainDlg.h

```
// FistPageMainDlg.h
//

#pragma once
#include "afxwin.h"

#include <opencv2/opencv.hpp>
#include"opencv/highgui.h"
#include"CvvImage.h"

using namespace cv;
```

```
// CFistPageMainDlg dialog
class CFistPageMainDlg : public CDialogEx
{
// contrctor
public:
    CFistPageMainDlg(CWnd* pParent = NULL);
    virtual INT_PTR DoModal();;
//data of dialog
#ifdef AFX_DESIGN_TIME
    //enum { IDD = IDD_FISTPAGEMAIN_DIALOG };
    enum { IDD = IDD_DIALOG_SECOND };
#endif


    protected:
    virtual void DoDataExchange(CDataExchange* pDX);


protected:
    HICON m_hIcon;


    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg void OnBnClickedButtonLoad();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);

    DECLARE_MESSAGE_MAP()
public:

    void DrawPicToHDC(char * pathfile,  UINT ID);//load image to widget
    bool IsControlClick(CPoint point,UINT ID);//judge the mouse in or not in
the widget
    void InitFistPictureShow();//showing iamge in the main page
    void ShowPicture(int num);//display image
    CString OpenandChooseFileFolder();//open the foler choosing dialog
    CString readInfoText(CString filepath);//read text file

public:
     int  ID_SHOW_PICTURE ;
     String szDir;//default directory
```

```
     afx_msg void OnStnClickedPicturePersoninfo();
     afx_msg void OnStnClickedPicture2weima();
     afx_msg void OnStnClickedPictureShow();
     afx_msg void OnStnClickedPicturePdf();
     afx_msg void OnStnClickedPictureDown();
};
```

MenuDlg.h

```
#pragma once


#include <opencv2/opencv.hpp>
#include"opencv/highgui.h"
#include"CvvImage.h"
#include"resource.h"
#include"SVRead.h"
#include"MyOpenGLView1.h"
#include"DialogGl.h"
class MenuDlg : public CDialog
{
    DECLARE_DYNAMIC(MenuDlg)

public:
    MenuDlg(CWnd* pParent = NULL);
    virtual ~MenuDlg();


#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_DIALOG1 };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    DECLARE_MESSAGE_MAP()
public:

    afx_msg void OnElectronMeanenergy();
    afx_msg void OnElectronMeanenergydensity();
    afx_msg void OnElectronElasticenergylosses();
    afx_msg void OnElectronElasticenergylinearisedlosses();
```

```
afx_msg void OnElectronMea();
afx_msg void OnElectronMeanenergylinearisedsource();
afx_msg void OnElectronMeanenergyfluxdensity();
afx_msg void OnElectronMeanenergyfluxdensity32782();
afx_msg void OnElectronReducedfield();
afx_msg void OnElectronReducedfield32784();
afx_msg void OnElectronEnergytable();
afx_msg void OnElectronElasticlosstable();




afx_msg void OnEmregionPotential();
afx_msg void OnEmregionCurrentdensity();
afx_msg void OnEmregionEx();
afx_msg void OnEmregionEy();
afx_msg void OnEmregionPowerdissipation();
afx_msg void OnEmregionV();
afx_msg void OnEmregionVolumechargedensity();
afx_msg void OnEmregionVolumechargedensity32902();
afx_msg void OnGeneralConfiguration();
afx_msg void OnGeometryConfiguration();
afx_msg void OnGeometryCvvolumes();
afx_msg void OnGeometryCvareas();
afx_msg void OnGeometryCvareas32871();

//afx_msg void OnLinechartDt();
afx_msg void OnLinechartEr();
afx_msg void OnLinechartEpsilon();
afx_msg void OnLinechartN0();
afx_msg void OnLinechartN1();
afx_msg void OnLinechartN2();
afx_msg void OnLinechartN3();
afx_msg void OnLinechartN4();
afx_msg void OnLinechartN5();
afx_msg void OnLinechartElasloss();
afx_msg void OnLinechartR0();
afx_msg void OnLinechartR1();
afx_msg void OnLinechartR2();
afx_msg void OnLinechartR3();
afx_msg void OnLinechartR4();

afx_msg void OnLinechartR5();
```

```
afx_msg void OnLinechartR6();
afx_msg void OnLinechartR7();
afx_msg void OnLinechartV1();
afx_msg void OnLinechartV2();
afx_msg void OnLinechartV4();
afx_msg void OnLinechartI1();
afx_msg void OnLinechartIc1();
afx_msg void OnLinechartI2();
afx_msg void OnLinechartIc2();
afx_msg void OnLinechartI4();
afx_msg void OnLinechartIc4();
afx_msg void OnGastemperatureGastemperature();
afx_msg void OnReactionsE();
afx_msg void OnReactionsE32789();
afx_msg void OnReactionsE32790();
afx_msg void OnReactionsE32791();
afx_msg void OnReactionsE32792();
afx_msg void OnReactionsE32793();
afx_msg void OnReactionsHe();
afx_msg void OnReactionsHe32795();
afx_msg void OnReactionsHe32796();
afx_msg void OnReactionsHe32797();
afx_msg void OnReactionsHe32798();
afx_msg void OnReactionsHe32799();
afx_msg void OnReactionsHe32800();
afx_msg void OnReactionsHe32801();
afx_msg void OnReactionsHe2();
afx_msg void OnReactionsHe3();
afx_msg void OnHeHe();
afx_msg void OnHeHe32810();
afx_msg void OnHeHe32811();
afx_msg void OnHeHe32812();
afx_msg void OnHeHe32813();
afx_msg void OnHeHe32814();
afx_msg void OnHeHe32815();
afx_msg void OnHeHe32816();
afx_msg void OnHeHe32817();
afx_msg void OnHeHe32818();
afx_msg void OnHeHe32819();
afx_msg void OnHeHe32820();
afx_msg void OnHeHe32821();
afx_msg void OnHeHe32822();
afx_msg void OnHeHe32823();
```

```
    afx_msg void OnHeHe32824();
    afx_msg void OnHeHe32825();
    afx_msg void OnHeHe32826();
    afx_msg void OnHe2He2();
    afx_msg void OnHe2He3();
    afx_msg void OnHe2He4();
    afx_msg void OnHe2He5();
    afx_msg void OnHe2He6();
    afx_msg void OnHe2He7();
    afx_msg void OnHe2He8();
    afx_msg void OnHe2He9();
    afx_msg void OnHe2He10();
    afx_msg void OnEEdensity();
    afx_msg void OnEEdiffusioncoef();
    afx_msg void OnEEmobility();
    afx_msg void OnEEsource();
    afx_msg void OnEElinearisedsource();
    afx_msg void OnEEfluxx();
    afx_msg void OnEEfluxy();
    afx_msg void OnEEpowerdissipation();
    afx_msg void OnEEmobilitytable();
    afx_msg void OnEE();
    afx_msg void OnEE32846();
    afx_msg void OnEE32847();
    afx_msg void OnEE32848();
    afx_msg void OnEE32849();
    afx_msg void OnEE32850();
    afx_msg void OnEE32851();
    afx_msg void OnEE32852();
    afx_msg void OnEE32853();
public:

    CString m_OpenFileFolder;
    CString m_EditShowTxt;//Display the data in text file
    CString **m_FileName;//Read the name from csv file

public:
    CMyOpenGLView1 *m_diaplay;
    virtual BOOL OnInitDialog();
    CString  GetExtensionOfFile(CString filePath);//return the type of file
based on suffix
    CString readInfoText(CString filepath);//Read the text file based on name
    CString ** ReadCsvFile(CString filepath);//Read the data in the csv file
```

```
    void  ShowFile(int id);//display different data of text file
    void DrawPicToHDC(char * pathfile, UINT ID);//loading image
    void ShowLine(int biaohao);

    afx_msg void OnLinechartQ1();
    afx_msg void OnLinechartQ2();
    afx_msg void OnLinechartQ4();
    afx_msg void OnLinechartP0();
    afx_msg void OnEnChangeEditShowtxt();
};
```

MyOpenGLView.h

```
#pragma once
#pragma comment(lib,"GL//opengl32.lib")
#pragma comment(lib,"GL//glut32.lib")
#pragma comment(lib,"GL//glu32.lib")
#pragma comment( lib, "GL//glaux.lib")

#include "afxwin.h"
#include"GL/GL.H"
#include"GL/glut.h"
#include"GL/GLU.H"
#include"SVRead.h"
class CMyOpenGLView :
    public CWnd
{
public:
    CMyOpenGLView(void);
    ~CMyOpenGLView(void);


public:
    void OnPaint2(int type);
    CPoint pointcurrent;
    BOOL lbuttondown;
    BOOL SetupLighting();
    BOOL SetupViewingTransform();
    BOOL SetupViewingFrustum(GLdouble aspect_ratio);
    BOOL SetupViewPort(int cx,int cy);
    BOOL SetupPixelFormat(HDC hdc);
    HGLRC hglrc;
```

```
    void drawrobot(int type);
    int drawrobottype;
    void ReadModel();
    void DrawTu1();
    void DrawTu2();
    bool m_dis1;
    bool m_dis2;
    void getdc();

protected:
    afx_msg void OnTimer(UINT_PTR nIDEvent);
    afx_msg int  OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
    afx_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);
    HDC hdc;
    DECLARE_MESSAGE_MAP()

public:
    afx_msg void OnDestroy();
public:
    CString **data;
    void ReadFile(CString pathfile, char divided);//Read file and get data
    float* ShowLine(int biaohao,float*kedu);
    int  data_row, data_col,biaohao_show;
    float minx, miny, maxx, maxy;
    float  gl_width, gl_heigth;
};
```

MyOpenGLView1.h

```
#pragma once

#pragma comment(lib,"GL//opengl32.lib")
#pragma comment(lib,"GL//glut32.lib")
#pragma comment(lib,"GL//glu32.lib")
#pragma comment( lib, "GL//glaux.lib")

#include "afxwin.h"
```

```
#include"GL/GL.H"
#include"GL/glut.h"
#include"GL/GLU.H"
#include"SVRead.h"
class CMyOpenGLView1 :
    public CWnd
{
public:
    CMyOpenGLView1(void);
    ~CMyOpenGLView1(void);


public:
    void OnPaint2(int type);
    CPoint pointcurrent;
    BOOL lbuttondown;
    BOOL SetupLighting();
    BOOL SetupViewingTransform();
    BOOL SetupViewingFrustum(GLdouble aspect_ratio);
    BOOL SetupViewPort(int cx,int cy);
    BOOL SetupPixelFormat(HDC hdc);
    HGLRC hglrc;

    void drawrobot(int type);
    int drawrobottype;
    void Readtxt(const char* filePath);
    void DrawTu1();
    void DrawTu2();

    void getdc();

protected:
    afx_msg void OnTimer(UINT_PTR nIDEvent);
    afx_msg int  OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
    afx_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);
    HDC hdc;
    DECLARE_MESSAGE_MAP()
```

```
public:
    afx_msg void OnDestroy();
public:


    float  gl_width, gl_heigth;
    afx_msg void OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnKillFocus(CWnd* pNewWnd);
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
};
struct HangVert
{
    float colorpoint[60];
};
```

MyPropertySheet.h

```
#pragma once



// MyPropertySheet

class MyPropertySheet : public CPropertySheet
{
    DECLARE_DYNAMIC(MyPropertySheet)

public:
    MyPropertySheet(UINT nIDCaption, CWnd* pParentWnd = NULL, UINT iSelectPage
= 0);
    MyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd = NULL, UINT
iSelectPage = 0);
    virtual ~MyPropertySheet();

protected:
    DECLARE_MESSAGE_MAP()
public:
    virtual BOOL OnInitDialog();
};
```

Page1.h

```
#pragma once

#include <opencv2/opencv.hpp>
#include"opencv/highgui.h"
#include"CvvImage.h"
#include"resource.h"


class CPage1 : public CPropertyPage
{
    DECLARE_DYNAMIC(CPage1)

public:
    CPage1();
    virtual ~CPage1();

#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_PAGE1 };
#endif

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

    DECLARE_MESSAGE_MAP()
public:

public:
    bool readInfoText(CString filepath);//Read file based on address
    void  Initial();
    void DrawPicToHDC(char * pathfile, UINT ID);//load picture
    CString m_editText;
    afx_msg void OnPaint();
    afx_msg void OnEnChangeEdit1();
};
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by FistPageMain.rc
//
#define IDM_ABOUTBOX                    0x0010
```

```
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_FISTPAGEMAIN_DIALOG         102
#define IDD_DIALOG_SECOND               102
#define IDR_MAINFRAME                   128
#define IDD_PAGE1                       130
#define IDR_MENU_PAGE2                  137
#define IDD_DIALOG1                     138
#define IDD_DIALOG_GL                   141
#define IDC_Pitcture_one                1002
#define IDC_PICTURE_2WEIMA              1004
#define IDC_PICTURE_PERSONINFO          1005
#define IDC_PICTURE_SHOW                1006
#define IDC_PICTURE_UP                  1007
#define IDC_PICTURE_DOWN                1008
#define IDC_PICTURE_PDF                 1009
#define IDC_BUTTON_LOAD                 1010
#define IDC_EDIT1                       1014
#define IDC_PICTURE1_BACKGROUND         1016
#define IDC_OCX1                        1018
#define IDC_EDIT_SHOWTXT                1019
#define IDC_PICTURE_DATA                1020
#define IDC_STATIC_OPENGL               1021
#define IDC_STATIC_GL                   1022
#define IDC_PICTURE_X                   1023
#define IDC_PICTURE_Y                   1024
#define IDC_STATIC_x1                   1025
#define IDC_STATIC_x2                   1026
#define IDC_STATIC_x3                   1027
#define IDC_STATIC_x4                   1028
#define IDC_STATIC_x5                   1029
#define IDC_STATIC_y1                   1030
#define IDC_STATIC_y2                   1031
#define IDC_STATIC_y3                   1032
#define IDC_STATIC_y4                   1033
#define IDC_STATIC_y5                   1034
#define IDC_BUTTON1                     1035
#define IDC_STATIC_y6                   1036
#define IDC_STATIC_x6                   1037
#define IDC_GL_STATIC                   1038
#define IDC_STATIC_GLTu                 1039
#define IDC_STATIC_name                 1040
#define ID_DISCHARGEREGION_ELECTRON     32771
```

```
#define ID_DISCHARGEREGION_GASTEMPERATURE 32772
#define ID_DISCHARGEREGION_REACTIONS    32773
#define ID_DISCHARGEREGION_SPECIES      32774
#define ID_ELECTRON_MEANENERGY          32775
#define ID_ELECTRON_MEANENERGYDENSITY   32776
#define ID_ELECTRON_ELASTICENERGYLOSSES 32777
#define ID_ELECTRON_ELASTICENERGYLINEARISEDLOSSES 32778
#define ID_ELECTRON_MEA                 32779
#define ID_ELECTRON_MEANENERGYLINEARISEDSOURCE 32780
#define ID_ELECTRON_MEANENERGYFLUXDENSITY 32781
#define ID_ELECTRON_MEANENERGYFLUXDENSITY32782 32782
#define ID_ELECTRON_REDUCEDFIELD        32783
#define ID_ELECTRON_REDUCEDFIELD32784   32784
#define ID_ELECTRON_ENERGYTABLE         32785
#define ID_ELECTRON_ELASTICLOSSTABLE    32786
#define ID_GASTEMPERATURE_GASTEMPERATURE 32787
#define ID_REACTIONS_E                  32788
#define ID_REACTIONS_E32789             32789
#define ID_REACTIONS_E32790             32790
#define ID_REACTIONS_E32791             32791
#define ID_REACTIONS_E32792             32792
#define ID_REACTIONS_E32793             32793
#define ID_REACTIONS_HE                 32794
#define ID_REACTIONS_HE32795            32795
#define ID_REACTIONS_HE32796            32796
#define ID_REACTIONS_HE32797            32797
#define ID_REACTIONS_HE32798            32798
#define ID_REACTIONS_HE32799            32799
#define ID_REACTIONS_HE32800            32800
#define ID_REACTIONS_HE32801            32801
#define ID_REACTIONS_HE2                32802
#define ID_REACTIONS_HE3                32803
#define ID_SPECIES_HE                   32804
#define ID_SPECIES_HE32805              32805
#define ID_SPECIES_HE2                  32806
#define ID_SPECIES_E                    32807
#define ID_SPECIES_E32808               32808
#define ID_HE_HE                        32809
#define ID_HE_HE32810                   32810
#define ID_HE_HE32811                   32811
#define ID_HE_HE32812                   32812
#define ID_HE_HE32813                   32813
#define ID_HE_HE32814                   32814
```

```
#define ID_HE_HE32815                32815
#define ID_HE_HE32816                32816
#define ID_HE_HE32817                32817
#define ID_HE_HE32818                32818
#define ID_HE_HE32819                32819
#define ID_HE_HE32820                32820
#define ID_HE_HE32821                32821
#define ID_HE_HE32822                32822
#define ID_HE_HE32823                32823
#define ID_HE_HE32824                32824
#define ID_HE_HE32825                32825
#define ID_HE_HE32826                32826
#define ID_HE2_HE2                   32827
#define ID_HE2_HE3                   32828
#define ID_HE2_HE4                   32829
#define ID_HE2_HE5                   32830
#define ID_HE2_HE6                   32831
#define ID_HE2_HE7                   32832
#define ID_HE2_HE8                   32833
#define ID_HE2_HE9                   32834
#define ID_HE2_HE10                  32835
#define ID_E_EDENSITY                32836
#define ID_E_EDIFFUSIONCOEF           32837
#define ID_E_EMOBILITY               32838
#define ID_E_ESOURCE                 32839
#define ID_E_ELINEARISEDSOURCE        32840
#define ID_E_EFLUXX                  32841
#define ID_E_EFLUXY                  32842
#define ID_E_EPOWERDISSIPATION        32843
#define ID_E_EMOBILITYTABLE          32844
#define ID_E_E                       32845
#define ID_E_E32846                  32846
#define ID_E_E32847                  32847
#define ID_E_E32848                  32848
#define ID_E_E32849                  32849
#define ID_E_E32850                  32850
#define ID_E_E32851                  32851
#define ID_E_E32852                  32852
#define ID_E_E32853                  32853
#define ID_EMREGION                  32854
#define ID_EMREGION_POTENTIAL         32855
#define ID_EMREGION_CURRENTDENSITY    32856
#define ID_EMREGION_EX               32857
```

```
#define ID_EMREGION_EY                 32858
#define ID_EMREGION_POWERDISSIPATION    32859
#define ID_EMREGION_V                  32860
#define ID_EMREGION_VOLUMECHARGEDENSITY 32861
#define ID_EMREGION_VOLUMECHARGEDENSITY32862 32862
#define ID_GENERAL                     32863
#define ID_GENERAL_CONFIGURATION        32864
#define ID_GENERAL_CVVOLUMES            32865
#define ID_GENERAL_CVAREAS              32866
#define ID_GENERAL_CVAREAS32867         32867
#define ID_GEOMETRY_CONFIGURATION       32868
#define ID_GEOMETRY_CVVOLUMES           32869
#define ID_GEOMETRY_CVAREAS             32870
#define ID_GEOMETRY_CVAREAS32871        32871
#define ID_LINECHART_DT                32872
#define ID_LINECHART_ER                32873
#define ID_LINECHART_EPSILON            32874
#define ID_LINECHART_N0                32875
#define ID_LINECHART_N1                32876
#define ID_LINECHART_N2                32877
#define ID_LINECHART_N3                32878
#define ID_LINECHART_N4                32879
#define ID_LINECHART_N5                32880
#define ID_LINECHART_ELASLOSS           32881
#define ID_LINECHART_R0                32882
#define ID_LINECHART_R1                32883
#define ID_LINECHART_R2                32884
#define ID_LINECHART_R3                32885
#define ID_LINECHART_R5                32886
#define ID_LINECHART_R6                32887
#define ID_LINECHART_R7                32888
#define ID_LINECHART_V1                32889
#define ID_LINECHART_V2                32890
#define ID_LINECHART_V4                32891
#define ID_LINECHART_I1                32892
#define ID_LINECHART_IC1               32893
#define ID_LINECHART_I2                32894
#define ID_LINECHART_IC2               32895
#define ID_LINECHART_I4                32896
#define ID_LINECHART_IC4               32897
#define ID_LINECHART_Q1                32898
#define ID_LINECHART_Q2                32899
#define ID_LINECHART_Q4                32900
```

```
#define ID_LINECHART_P0                 32901
#define ID_EMREGION_VOLUMECHARGEDENSITY32902 32902
#define ID_LINECHART_R4                 32903

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        144
#define _APS_NEXT_COMMAND_VALUE         32904
#define _APS_NEXT_CONTROL_VALUE         1041
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

stdafx.h

```
#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN
#endif

#include "targetver.h"

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS


#define _AFX_ALL_WARNINGS

#include <afxwin.h>
#include <afxext.h>


#include <afxdisp.h>



#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
```

```
#include <afxcmn.h>
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxcontrolbars.h>
#include <afxdlgs.h>
#include <afxdisp.h>









#ifdef _UNICODE
#if defined _M_IX86
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='*'\"")
#elif defined _M_X64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df'
language='*'\"")
#else
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0'
processorArchitecture='*' publicKeyToken='6595b64144ccf1df' language='*'\"")
#endif
#endif
```

SVRead.h

```
#pragma once

class CSVRead
{
public:
    CSVRead(CString ReadFilePath);
    ~CSVRead(void);
    CStdioFile file;
protected:
```

```
    //Obtain data from CSV file
    CStringArray* DevideStr(CString ItemFieldStr , char divider);
public:
    // Read and save the data in CSV
    CStringArray * CSVInf(char divided);
    // obtain the rows of CSV
    int FileRowCount(void);

    CString DeleteBlank(CString str);//Transform continues space to single
    static CString ** ReadCsvFile(CString filepath, char divided, int *row, int
*col);//OUTPUT AN ARRARY
};
```

targetver.h

```
#pragma once
#include <SDKDDKVer.h>
```

Class file:

CvvImage.cpp

```
#include "StdAfx.h"
#include "CvvImage.h"
////////////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////////////
CV_INLINE RECT NormalizeRect( RECT r );
CV_INLINE RECT NormalizeRect( RECT r )
{
    int t;
    if( r.left > r.right )
    {
        t = r.left;
        r.left = r.right;
        r.right = t;
    }
    if( r.top > r.bottom )
    {
        t = r.top;
        r.top = r.bottom;
        r.bottom = t;
    }
```

```
    return r;
}
CV_INLINE CvRect RectToCvRect( RECT sr );
CV_INLINE CvRect RectToCvRect( RECT sr )
{
    sr = NormalizeRect( sr );
    return cvRect( sr.left, sr.top, sr.right - sr.left, sr.bottom - sr.top );
}
CV_INLINE RECT CvRectToRect( CvRect sr );
CV_INLINE RECT CvRectToRect( CvRect sr )
{
    RECT dr;
    dr.left = sr.x;
    dr.top = sr.y;
    dr.right = sr.x + sr.width;
    dr.bottom = sr.y + sr.height;

    return dr;
}
CV_INLINE IplROI RectToROI( RECT r );
CV_INLINE IplROI RectToROI( RECT r )
{
    IplROI roi;
    r = NormalizeRect( r );
    roi.xOffset = r.left;
    roi.yOffset = r.top;
    roi.width = r.right - r.left;
    roi.height = r.bottom - r.top;
    roi.coi = 0;

    return roi;
}
void  FillBitmapInfo( BITMAPINFO* bmi, int width, int height, int bpp, int
origin )
{
    assert( bmi && width >= 0 && height >= 0 && (bpp == 8 || bpp == 24 || bpp
== 32));

    BITMAPINFOHEADER* bmih = &(bmi->bmiHeader);

    memset( bmih, 0, sizeof(*bmih));
    bmih->biSize = sizeof(BITMAPINFOHEADER);
```

```
    bmih->biWidth = width;
    bmih->biHeight = origin ? abs(height) : -abs(height);
    bmih->biPlanes = 1;
    bmih->biBitCount = (unsigned short)bpp;
    bmih->biCompression = BI_RGB;
    if( bpp == 8 )
    {
        RGBQUAD* palette = bmi->bmiColors;
        int i;
        for( i = 0; i < 256; i++ )
        {
            palette[i].rgbBlue = palette[i].rgbGreen = palette[i].rgbRed =
(BYTE)i;
            palette[i].rgbReserved = 0;
        }
    }
}
CvvImage::CvvImage()
{
    m_img = 0;
}
void CvvImage::Destroy()
{
    cvReleaseImage( &m_img );
}
CvvImage::~CvvImage()
{
    Destroy();
}
bool  CvvImage::Create( int w, int h, int bpp, int origin )
{
    const unsigned max_img_size = 10000;

    if( (bpp != 8 && bpp != 24 && bpp != 32) ||
        (unsigned)w >=  max_img_size || (unsigned)h >= max_img_size ||
        (origin != IPL_ORIGIN_TL && origin != IPL_ORIGIN_BL))
    {
        assert(0); // most probably, it is a programming error
        return false;
    }
    if( !m_img || Bpp() != bpp || m_img->width != w || m_img->height != h )
    {
        if( m_img && m_img->nSize == sizeof(IplImage))
```

```
            Destroy();
        /* prepare IPL header */
        m_img = cvCreateImage( cvSize( w, h ), IPL_DEPTH_8U, bpp/8 );
    }
    if( m_img )
        m_img->origin = origin == 0 ? IPL_ORIGIN_TL : IPL_ORIGIN_BL;
    return m_img != 0;
}
void  CvvImage::CopyOf( CvvImage& image, int desired_color )
{
    IplImage* img = image.GetImage();
    if( img )
    {
        CopyOf( img, desired_color );
    }
}
#define HG_IS_IMAGE(img)\
    ((img) != 0 && ((const IplImage*)(img))->nSize == sizeof(IplImage) &&\
    ((IplImage*)img)->imageData != 0)
void  CvvImage::CopyOf( IplImage* img, int desired_color )
{
    if( HG_IS_IMAGE(img) )
    {
        int color = desired_color;
        CvSize size = cvGetSize( img );
        if( color < 0 )
            color = img->nChannels > 1;
        if( Create( size.width, size.height,
            (!color ? 1 : img->nChannels > 1 ? img->nChannels : 3)*8,
            img->origin ))
        {
            cvConvertImage( img, m_img, 0 );
        }
    }
}
bool  CvvImage::Load( const char* filename, int desired_color )
{
    IplImage* img = cvLoadImage( filename, desired_color );
    if( !img )
        return false;

    CopyOf( img, desired_color );
    cvReleaseImage( &img );
```

```
    return true;
}
bool  CvvImage::LoadRect( const char* filename,
    int desired_color, CvRect r )
{
    if( r.width < 0 || r.height < 0 ) return false;

    IplImage* img = cvLoadImage( filename, desired_color );
    if( !img )
        return false;
    if( r.width == 0 || r.height == 0 )
    {
        r.width = img->width;
        r.height = img->height;
        r.x = r.y = 0;
    }
    if( r.x > img->width || r.y > img->height ||
        r.x + r.width < 0 || r.y + r.height < 0 )
    {
        cvReleaseImage( &img );
        return false;
    }
    /* truncate r to source image */
    if( r.x < 0 )
    {
        r.width += r.x;
        r.x = 0;
    }
    if( r.y < 0 )
    {
        r.height += r.y;
        r.y = 0;
    }
    if( r.x + r.width > img->width )
        r.width = img->width - r.x;

    if( r.y + r.height > img->height )
        r.height = img->height - r.y;
    cvSetImageROI( img, r );
    CopyOf( img, desired_color );
    cvReleaseImage( &img );
    return true;
```

```
}
bool  CvvImage::Save( const char* filename )
{
    if( !m_img )
        return false;
    cvSaveImage( filename, m_img );
    return true;
}
void  CvvImage::Show( const char* window )
{
    if( m_img )
        cvShowImage( window, m_img );
}
void  CvvImage::Show( HDC dc, int x, int y, int w, int h, int from_x, int
from_y )
{
    if( m_img && m_img->depth == IPL_DEPTH_8U )
    {
        uchar buffer[sizeof(BITMAPINFOHEADER) + 1024];
        BITMAPINFO* bmi = (BITMAPINFO*)buffer;
        int bmp_w = m_img->width, bmp_h = m_img->height;
        FillBitmapInfo( bmi, bmp_w, bmp_h, Bpp(), m_img->origin );
        from_x = MIN( MAX( from_x, 0 ), bmp_w - 1 );
        from_y = MIN( MAX( from_y, 0 ), bmp_h - 1 );
        int sw = MAX( MIN( bmp_w - from_x, w ), 0 );
        int sh = MAX( MIN( bmp_h - from_y, h ), 0 );
        SetDIBitsToDevice(
            dc, x, y, sw, sh, from_x, from_y, from_y, sh,
            m_img->imageData + from_y*m_img->widthStep,
            bmi, DIB_RGB_COLORS );
    }
}
void  CvvImage::DrawToHDC( HDC hDCDst, RECT* pDstRect )
{
    if( pDstRect && m_img && m_img->depth == IPL_DEPTH_8U && m_img->imageData )
    {
        uchar buffer[sizeof(BITMAPINFOHEADER) + 1024];
        BITMAPINFO* bmi = (BITMAPINFO*)buffer;
        int bmp_w = m_img->width, bmp_h = m_img->height;
        CvRect roi = cvGetImageROI( m_img );
        CvRect dst = RectToCvRect( *pDstRect );
        if( roi.width == dst.width && roi.height == dst.height )
        {
```

```
            Show( hDCDst, dst.x, dst.y, dst.width, dst.height, roi.x, roi.y );
            return;
        }
        if( roi.width > dst.width )
        {
            SetStretchBltMode(
                hDCDst,          // handle to device context
                HALFTONE );
        }
        else
        {
            SetStretchBltMode(
                hDCDst,          // handle to device context
                COLORONCOLOR );
        }
        FillBitmapInfo( bmi, bmp_w, bmp_h, Bpp(), m_img->origin );
        ::StretchDIBits(
            hDCDst,
            dst.x, dst.y, dst.width, dst.height,
            roi.x, roi.y, roi.width, roi.height,
            m_img->imageData, bmi, DIB_RGB_COLORS, SRCCOPY );
    }
}
void  CvvImage::Fill( int color )
{
    cvSet( m_img,
cvScalar(color&255,(color>>8)&255,(color>>16)&255,(color>>24)&255) );
}
```

DialogGl.cpp

```
// DialogGl.cpp
//

#include "stdafx.h"
//#include "FistPageMain.h"
#include "DialogGl.h"
#include "afxdialogex.h"
#include"resource.h"

// DialogGl dialog

IMPLEMENT_DYNAMIC(DialogGl, CDialog)
```

```
DialogGl::DialogGl(CWnd* pParent /*=NULL*/) //constructor
    : CDialog(IDD_DIALOG_GL, pParent)
    , kedu_x1(_T(""))
    , kedu_x2(_T(""))
    , kedu_x3(_T(""))
    , kedu_x4(_T(""))
    , kedu_x5(_T(""))
    , kedu_y1(_T(""))
    , kedu_y2(_T(""))
    , kedu_y3(_T(""))
    , kedu_y4(_T(""))
    , kedu_y5(_T(""))
    , kedu_x6(_T(""))
    , kedu_y6(_T(""))
    , m_glName(_T(""))
    , m_name(_T(""))
{
    for (int i = 0; i < 4; i++)
        kedu[i] = 0;
}

DialogGl::~DialogGl()
{
}

void DialogGl::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_STATIC_x1, kedu_x1);
    DDX_Text(pDX, IDC_STATIC_x2, kedu_x2);
    DDX_Text(pDX, IDC_STATIC_x3, kedu_x3);
    DDX_Text(pDX, IDC_STATIC_x4, kedu_x4);
    DDX_Text(pDX, IDC_STATIC_x5, kedu_x5);
    DDX_Text(pDX, IDC_STATIC_y1, kedu_y1);
    DDX_Text(pDX, IDC_STATIC_y2, kedu_y2);
    DDX_Text(pDX, IDC_STATIC_y3, kedu_y3);
    DDX_Text(pDX, IDC_STATIC_y4, kedu_y4);
    DDX_Text(pDX, IDC_STATIC_y5, kedu_y5);

    DDX_Text(pDX, IDC_STATIC_x6, kedu_x6);
    DDX_Text(pDX, IDC_STATIC_y6, kedu_y6);
    DDX_Text(pDX, IDC_GL_STATIC, m_glName);
```

```
    DDX_Text(pDX, IDC_STATIC_name, m_name);
}


BEGIN_MESSAGE_MAP(DialogGl, CDialog)
    ON_BN_CLICKED(IDOK, &DialogGl::OnBnClickedOk)
    ON_BN_CLICKED(IDC_BUTTON1, &DialogGl::OnBnClickedButton1)
    ON_WM_PAINT()
    ON_STN_CLICKED(IDC_STATIC_x5, &DialogGl::OnStnClickedStaticx5)
    ON_STN_CLICKED(IDC_STATIC_y5, &DialogGl::OnStnClickedStaticy5)
    ON_STN_CLICKED(IDC_STATIC_y6, &DialogGl::OnStnClickedStaticy6)
    ON_STN_CLICKED(IDC_STATIC_GL, &DialogGl::OnStnClickedStaticGl)
END_MESSAGE_MAP()


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


    m_diaplay = new CMyOpenGLView();
    CRect  rect;
    GetDlgItem(IDC_STATIC_GL)->GetWindowRect(&rect);
    ScreenToClient(&rect);
    m_diaplay->Create(NULL, NULL, WS_VISIBLE, rect, this, 0);


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

}
void DialogGl::DrawPicToHDC(char * pathfile, UINT ID)
{

    IplImage *image = NULL; //Original image

    if (image) cvReleaseImage(&image);
    image = cvLoadImage(pathfile, 1); //Display image
    //Transfor this image to Cvvimage format
    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC = pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
```

```
    cimg.CopyOf(image); // copy this image
    cimg.DrawToHDC(hDC, &rect); // draw this image to widget
    ReleaseDC(pDC);
}
void DialogGl::ShowLine(int biaohao)
{
    for (int i = 0; i < 4; i++)
        kedu[i] = 0;
     m_diaplay->ShowLine(biaohao, kedu);
}
void DialogGl::ReadCsvFile(CString pathfile, char divided)
{

    m_diaplay->ReadFile(pathfile, divided);
}

void DialogGl::OnBnClickedOk()
{

    CDialog::OnOK();
}


void DialogGl::OnBnClickedButton1()
{

    //this->ShowWindow(false);
    CDialog::OnOK();
}


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



    DrawPicToHDC("externResouce\\FirstPicture\\x刻度.png", IDC_PICTURE_X);
    DrawPicToHDC("externResouce\\FirstPicture\\y刻度.png", IDC_PICTURE_Y);


        kedu_x1.Format("%.2e", kedu[0]);
        kedu_x2.Format("%.2e", kedu[0] + (kedu[1] - kedu[0]) / 5);
```

```
        kedu_x3.Format("%.2e", kedu[0] + 2 * (kedu[1] - kedu[0]) / 5);
        kedu_x4.Format("%.2e", kedu[0] + 3 * (kedu[1] - kedu[0]) / 5);
        kedu_x5.Format("%.2e", kedu[0] + 4 * (kedu[1] - kedu[0]) / 5);
        kedu_x6.Format("%.2e", kedu[0] + 5 * (kedu[1] - kedu[0]) / 5);
                                                                // Divide the
number to 6 areas to mark the scale
        kedu_y1.Format("%.2e", kedu[2]);
        kedu_y2.Format("%.2e", kedu[2] + (kedu[3] - kedu[2]) / 5);
        kedu_y3.Format("%.2e", kedu[2] + 2 * (kedu[3] - kedu[2]) / 5);
        kedu_y4.Format("%.2e", kedu[2] + 3 * (kedu[3] - kedu[2]) / 5);
        kedu_y5.Format("%.2e", kedu[2] + 4 * (kedu[3] - kedu[2]) / 5);
        kedu_y6.Format("%.2e", kedu[2] + 5 * (kedu[3] - kedu[2]) / 5);

        UpdateData(FALSE);

        int a = 0;
}


void DialogGl::OnStnClickedStaticx5()
{

}


void DialogGl::OnStnClickedStaticy5()
{

}


void DialogGl::OnStnClickedStaticy6()
{

}


void DialogGl::OnStnClickedStaticGl()
{

}
```

FirstPageMain.cpp

```
// FistPageMain.cpp :
//

#include "stdafx.h"
#include "FistPageMain.h"
#include "FistPageMainDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CFistPageMainApp

BEGIN_MESSAGE_MAP(CFistPageMainApp, CWinApp)
    ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()


// CFistPageMainApp

CFistPageMainApp::CFistPageMainApp()
{
    // support restart Root Explorer
    m_dwRestartManagerSupportFlags = AFX_RESTART_MANAGER_SUPPORT_RESTART;

}



CFistPageMainApp theApp;


// CFistPageMainApp initialization

BOOL CFistPageMainApp::InitInstance()
{

    INITCOMMONCONTROLSEX InitCtrls;
    InitCtrls.dwSize = sizeof(InitCtrls);
    // set to common widget
    InitCtrls.dwICC = ICC_WIN95_CLASSES;
```

```
    InitCommonControlsEx(&InitCtrls);

    CWinApp::InitInstance();


    AfxEnableControlContainer();


    CShellManager *pShellManager = new CShellManager;

    // active the management of vision
    CMFCVisualManager::SetDefaultManager(RUNTIME_CLASS(CMFCVisualManagerWindows
));


    SetRegistryKey(_T("应用程序向导生成的本地应用程序"));

    CFistPageMainDlg dlg;
    m_pMainWnd = &dlg;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {

    }
    else if (nResponse == IDCANCEL)
    {

    }
    else if (nResponse == -1)
    {
        TRACE(traceAppMsg, 0, "警告: 对话框创建失败，应用程序将意外终止。\n");
        TRACE(traceAppMsg, 0, "警告: 如果您在对话框上使用 MFC 控件，则无法 #define
_AFX_NO_MFC_CONTROLS_IN_DIALOGS。\n");
    }


    if (pShellManager != NULL)
    {
        delete pShellManager;
    }

#ifndef _AFXDLL
    ControlBarCleanUp();
```

```
#endif
    return FALSE;
}
```

FirstPageMainDlg.cpp

```
// FistPageMainDlg.cpp :
//

#include "stdafx.h"
#include "FistPageMain.h"
#include "FistPageMainDlg.h"
#include "afxdialogex.h"
#include"resource.h"
#include"Page1.h"
#include"MyPropertySheet.h"
#include"MenuDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif


//****************************** generate
automatic*****************************************//

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

// data of dialog
#ifdef AFX_DESIGN_TIME
    enum { IDD = IDD_ABOUTBOX };
#endif

    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support


protected:
    DECLARE_MESSAGE_MAP()
};
```

```
CAboutDlg::CAboutDlg() : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()


// CFistPageMainDlg dialog



CFistPageMainDlg::CFistPageMainDlg(CWnd* pParent /*=NULL*/)
    : CDialogEx(IDD_FISTPAGEMAIN_DIALOG, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

    ID_SHOW_PICTURE = 0;
}

void CFistPageMainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);




}

BEGIN_MESSAGE_MAP(CFistPageMainDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_LBUTTONDOWN()
    ON_BN_CLICKED(IDC_BUTTON_LOAD, &CFistPageMainDlg::OnBnClickedButtonLoad)
    ON_STN_CLICKED(IDC_PICTURE_PERSONINFO,
&CFistPageMainDlg::OnStnClickedPicturePersoninfo)
    ON_STN_CLICKED(IDC_PICTURE_2WEIMA,
```

```
&CFistPageMainDlg::OnStnClickedPicture2weima)
    ON_STN_CLICKED(IDC_PICTURE_SHOW,
&CFistPageMainDlg::OnStnClickedPictureShow)
    ON_STN_CLICKED(IDC_PICTURE_PDF, &CFistPageMainDlg::OnStnClickedPicturePdf)
    ON_STN_CLICKED(IDC_PICTURE_DOWN,
&CFistPageMainDlg::OnStnClickedPictureDown)
END_MESSAGE_MAP()


// CFistPageMainDlg information process

BOOL CFistPageMainDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();


    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

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


    SetIcon(m_hIcon, TRUE);
    SetIcon(m_hIcon, FALSE);

    return TRUE;
}

//****************************** generate
automatic*************************************//
```

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

// Add minimum button

INT_PTR CFistPageMainDlg::DoModal()
{

    return CDialogEx::DoModal();
}

void CFistPageMainDlg::OnPaint()
{
    InitFistPictureShow();//Showing image of main page

    if (IsIconic())
    {
        CPaintDC dc(this); // draw the device

        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

        // make sure the work space on the middle
        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
    {
        CDialogEx::OnPaint();
    }
}

//obtain the cursor when user dragging the minimum window

HCURSOR CFistPageMainDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

//click“Load”
void CFistPageMainDlg::OnBnClickedButtonLoad()
{
    // TODO: 在此添加控件通知处理程序代码
    MyPropertySheet sh(_T("Plasimo Display Assistant"), this, 2); // Default
display in third page
    CPage1 p1;

    // add one page
    sh.AddPage(&p1);



    CString szDir = OpenandChooseFileFolder();//Open infor.out
    if (szDir == "")
        return;
    CString infor_text = readInfoText(szDir);//Read infor.out
    if (infor_text == "")
        return;

    p1.m_editText = infor_text;
    sh.DoModal();             //information display

    //display the menu
    MenuDlg Menudlg;
    Menudlg.m_OpenFileFolder = szDir;
    Menudlg.DoModal();
}
//Click action
void CFistPageMainDlg::OnLButtonDown(UINT nFlags, CPoint point)
```

```
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值

    if (IsControlClick(point, IDC_PICTURE_2WEIMA))//click QR image
    {
            ShellExecute(NULL, _T("open"),
_T("http://fypforjunmingzhang.blogspot.co.uk/"), NULL, NULL, SW_SHOW);
    }

    if (IsControlClick(point, IDC_PICTURE_UP))//click < button
    {



        if (--ID_SHOW_PICTURE <= 0)
            ID_SHOW_PICTURE = 0;
        ShowPicture(ID_SHOW_PICTURE);



    }
    if (IsControlClick(point, IDC_PICTURE_DOWN))//click > button
    {
        if (++ID_SHOW_PICTURE >= 2)
            ID_SHOW_PICTURE = 2;

        ShowPicture(ID_SHOW_PICTURE);

    }
    if (IsControlClick(point, IDC_PICTURE_PDF))//click help button
    {
                TCHAR szFilter[] = _T("文本文件(*.pdf)|*.pdf|");
                CFileDialog fileDlg(TRUE, _T("txt"), NULL, 0, szFilter, this);
                CString strFilePath;

                {
                    system("ProgramSpecification.pdf");
                    system("close");
                    // if click help button, it will onpen
"programSpecification" file
                }
    }
    CDialogEx::OnLButtonDown(nFlags, point);//realize
}
```

```
//////////-------------------------------------------------------------------
-----------------------------
//loading images
void CFistPageMainDlg::DrawPicToHDC(char * pathfile, UINT ID)
{

    IplImage *image = NULL; //initial image
    if (image) cvReleaseImage(&image);

    image = cvLoadImage(pathfile, 1); //display image

    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC = pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
    cimg.CopyOf(image); // copy image
    cimg.DrawToHDC(hDC, &rect); // move this image to right widget area
    ReleaseDC(pDC);
}
//To judge the mouse area weather into the widget
bool CFistPageMainDlg::IsControlClick(CPoint point, UINT ID)
{
    CRect rect;
    GetDlgItem(ID)->GetWindowRect(&rect);
    ScreenToClient(rect);
    CPoint pt = (rect.TopLeft());

    GetDlgItem(ID)->GetClientRect(&rect);

    if (point.x > pt.x && point.x < (int)pt.x + rect.Width() &&
        point.y > pt.y && point.y < (int)pt.y + rect.Height()
        )
    {

        return true;
    }

    return  false;

}
//open the choose folder function
CString  CFistPageMainDlg::OpenandChooseFileFolder()//load file
```

```
{
    TCHAR           szFolderPath[MAX_PATH] = { 0 };
    CString         strFolderPath = TEXT("");

     BROWSEINFO      sInfo;
    ::ZeroMemory(&sInfo, sizeof(BROWSEINFO));
    sInfo.pidlRoot = 0;
    sInfo.lpszTitle = _T("Please choose the data folder : ");
    sInfo.ulFlags = BIF_DONTGOBELOWDOMAIN | BIF_RETURNONLYFSDIRS |
BIF_NEWDIALOGSTYLE | BIF_EDITBOX;
    sInfo.lpfn = NULL;

    // display the dialog of choosing a folder
    LPITEMIDLIST lpidlBrowse = ::SHBrowseForFolder(&sInfo);
    if (lpidlBrowse != NULL)
    {
        // get the name of folder
        if (::SHGetPathFromIDList(lpidlBrowse, szFolderPath))
        {
            strFolderPath = szFolderPath;
        }
    }
    if (lpidlBrowse != NULL)
    {
        ::CoTaskMemFree(lpidlBrowse);
    }

    return   strFolderPath;
}
//Read text files
CString  CFistPageMainDlg::readInfoText(CString filepath)//find the folder
address and read text files
{

    if (!PathFileExists(filepath + "\\" + "info.out"))
        return "";


    CFile   file(filepath + "\\" + "info.out", CFile::modeRead);//open
information text only for read
    char   *pBuf;
    int   iLen = file.GetLength(); //get the length of text
    pBuf = new   char[iLen + 1];
```

```
    file.Read(pBuf, iLen);
    pBuf[iLen] = 0;//relise memory
    file.Close();

    CString A; A.Format("%s", pBuf);

    free(pBuf);

    return  A;
}
//--------------------------------------------------------------------------------
-------------
//Display initial iamges
void CFistPageMainDlg::InitFistPictureShow()//initialization
{
    DrawPicToHDC("externResouce\\FirstPicture\\Background.png",
IDC_Pitcture_one);//background image
    DrawPicToHDC("externResouce\\FirstPicture\\91-100.png",
IDC_PICTURE_2WEIMA);//QR code
    DrawPicToHDC("externResouce\\FirstPicture\\50-125.png",
IDC_PICTURE_PERSONINFO);//my contact way
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-up.png",
IDC_PICTURE_UP);//< button
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-down.png",
IDC_PICTURE_DOWN);//> button
    DrawPicToHDC("externResouce\\FirstPicture\\25-25-PDF.png",
IDC_PICTURE_PDF);//help button

    DrawPicToHDC("externResouce\\FirstPicture\\FirstPicture.JPG",
IDC_PICTURE_SHOW);//background image
}
void CFistPageMainDlg::ShowPicture(int num)
{
    switch (num)
    {

    case 0:
        DrawPicToHDC("externResouce\\FirstPicture\\FirstPicture.JPG",
IDC_PICTURE_SHOW);//first image
        break;
    case 1:
        DrawPicToHDC("externResouce\\FirstPicture\\SecondPicture.JPG",
IDC_PICTURE_SHOW);//second image
```

```
        break;
    case 2:
        DrawPicToHDC("externResouce\\FirstPicture\\ThirdPicture.JPG",
IDC_PICTURE_SHOW);//third image
        break;
    default:

        break;
    }

}




void CFistPageMainDlg::OnStnClickedPicturePersoninfo()
{

}


void CFistPageMainDlg::OnStnClickedPicture2weima()
{

}


void CFistPageMainDlg::OnStnClickedPictureShow()
{

}


void CFistPageMainDlg::OnStnClickedPicturePdf()
{

}


void CFistPageMainDlg::OnStnClickedPictureDown()
{

}
```

MenuDlg.cpp

```
#include "stdafx.h"
#include "FistPageMain.h"
#include "MenuDlg.h"
#include "afxdialogex.h"

// MenuDlg

IMPLEMENT_DYNAMIC(MenuDlg, CDialog)

MenuDlg::MenuDlg(CWnd* pParent /*=NULL*/)
    : CDialog(IDD_DIALOG1, pParent),m_OpenFileFolder("")
    , m_EditShowTxt(_T(""))
{

}

MenuDlg::~MenuDlg()
{
}

void MenuDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);

    //DDX_Control(pDX, IDC_OCX1, m_Player);
    DDX_Text(pDX, IDC_EDIT_SHOWTXT, m_EditShowTxt);
}


BEGIN_MESSAGE_MAP(MenuDlg, CDialog)
    ON_COMMAND(ID_ELECTRON_MEANENERGY, &MenuDlg::OnElectronMeanenergy)
    ON_COMMAND(ID_ELECTRON_MEANENERGYDENSITY,
&MenuDlg::OnElectronMeanenergydensity)
    ON_COMMAND(ID_ELECTRON_ELASTICENERGYLOSSES,
&MenuDlg::OnElectronElasticenergylosses)
    ON_COMMAND(ID_ELECTRON_ELASTICENERGYLINEARISEDLOSSES,
&MenuDlg::OnElectronElasticenergylinearisedlosses)
    ON_COMMAND(ID_ELECTRON_MEA, &MenuDlg::OnElectronMea)
    ON_COMMAND(ID_ELECTRON_MEANENERGYLINEARISEDSOURCE,
```

```
&MenuDlg::OnElectronMeanenergylinearisedsource)
    ON_COMMAND(ID_ELECTRON_MEANENERGYFLUXDENSITY,
&MenuDlg::OnElectronMeanenergyfluxdensity)
    ON_COMMAND(ID_ELECTRON_MEANENERGYFLUXDENSITY32782,
&MenuDlg::OnElectronMeanenergyfluxdensity32782)
    ON_COMMAND(ID_ELECTRON_REDUCEDFIELD, &MenuDlg::OnElectronReducedfield)
    ON_COMMAND(ID_ELECTRON_REDUCEDFIELD32784,
&MenuDlg::OnElectronReducedfield32784)
    ON_COMMAND(ID_ELECTRON_ENERGYTABLE, &MenuDlg::OnElectronEnergytable)
    ON_COMMAND(ID_ELECTRON_ELASTICLOSSTABLE,
&MenuDlg::OnElectronElasticlosstable)
    ON_COMMAND(ID_EMREGION_POTENTIAL, &MenuDlg::OnEmregionPotential)
    ON_COMMAND(ID_EMREGION_CURRENTDENSITY, &MenuDlg::OnEmregionCurrentdensity)
    ON_COMMAND(ID_EMREGION_EX, &MenuDlg::OnEmregionEx)
    ON_COMMAND(ID_EMREGION_EY, &MenuDlg::OnEmregionEy)
    ON_COMMAND(ID_EMREGION_POWERDISSIPATION,
&MenuDlg::OnEmregionPowerdissipation)
    ON_COMMAND(ID_EMREGION_V, &MenuDlg::OnEmregionV)
    ON_COMMAND(ID_EMREGION_VOLUMECHARGEDENSITY,
&MenuDlg::OnEmregionVolumechargedensity)
    ON_COMMAND(ID_EMREGION_VOLUMECHARGEDENSITY32902,
&MenuDlg::OnEmregionVolumechargedensity32902)
    ON_COMMAND(ID_GENERAL_CONFIGURATION, &MenuDlg::OnGeneralConfiguration)
    ON_COMMAND(ID_GEOMETRY_CONFIGURATION, &MenuDlg::OnGeometryConfiguration)
    ON_COMMAND(ID_GEOMETRY_CVVOLUMES, &MenuDlg::OnGeometryCvvolumes)
    ON_COMMAND(ID_GEOMETRY_CVAREAS, &MenuDlg::OnGeometryCvareas)
    ON_COMMAND(ID_GEOMETRY_CVAREAS32871, &MenuDlg::OnGeometryCvareas32871)
    //ON_COMMAND(ID_LINECHART_DT, &MenuDlg::OnLinechartDt)
    ON_COMMAND(ID_LINECHART_ER, &MenuDlg::OnLinechartEr)
    ON_COMMAND(ID_LINECHART_EPSILON, &MenuDlg::OnLinechartEpsilon)
    ON_COMMAND(ID_LINECHART_N0, &MenuDlg::OnLinechartN0)
    ON_COMMAND(ID_LINECHART_N1, &MenuDlg::OnLinechartN1)
    ON_COMMAND(ID_LINECHART_N2, &MenuDlg::OnLinechartN2)
    ON_COMMAND(ID_LINECHART_N3, &MenuDlg::OnLinechartN3)
    ON_COMMAND(ID_LINECHART_N4, &MenuDlg::OnLinechartN4)
    ON_COMMAND(ID_LINECHART_N5, &MenuDlg::OnLinechartN5)
    ON_COMMAND(ID_LINECHART_ELASLOSS, &MenuDlg::OnLinechartElasloss)
    ON_COMMAND(ID_LINECHART_R0, &MenuDlg::OnLinechartR0)
    ON_COMMAND(ID_LINECHART_R1, &MenuDlg::OnLinechartR1)
    ON_COMMAND(ID_LINECHART_R2, &MenuDlg::OnLinechartR2)
    ON_COMMAND(ID_LINECHART_R3, &MenuDlg::OnLinechartR3)
    ON_COMMAND(ID_LINECHART_R4, &MenuDlg::OnLinechartR4)
    ON_COMMAND(ID_LINECHART_R5, &MenuDlg::OnLinechartR5)
```

```
    ON_COMMAND(ID_LINECHART_R6, &MenuDlg::OnLinechartR6)
    ON_COMMAND(ID_LINECHART_R7, &MenuDlg::OnLinechartR7)
    ON_COMMAND(ID_LINECHART_V1, &MenuDlg::OnLinechartV1)
    ON_COMMAND(ID_LINECHART_V2, &MenuDlg::OnLinechartV2)
    ON_COMMAND(ID_LINECHART_V4, &MenuDlg::OnLinechartV4)
    ON_COMMAND(ID_LINECHART_I1, &MenuDlg::OnLinechartI1)
    ON_COMMAND(ID_LINECHART_IC1, &MenuDlg::OnLinechartIc1)
    ON_COMMAND(ID_LINECHART_I2, &MenuDlg::OnLinechartI2)
    ON_COMMAND(ID_LINECHART_IC2, &MenuDlg::OnLinechartIc2)
    ON_COMMAND(ID_LINECHART_I4, &MenuDlg::OnLinechartI4)
    ON_COMMAND(ID_LINECHART_IC4, &MenuDlg::OnLinechartIc4)
    ON_COMMAND(ID_GASTEMPERATURE_GASTEMPERATURE,
&MenuDlg::OnGastemperatureGastemperature)
    ON_COMMAND(ID_REACTIONS_E, &MenuDlg::OnReactionsE)
    ON_COMMAND(ID_REACTIONS_E32789, &MenuDlg::OnReactionsE32789)
    ON_COMMAND(ID_REACTIONS_E32790, &MenuDlg::OnReactionsE32790)
    ON_COMMAND(ID_REACTIONS_E32791, &MenuDlg::OnReactionsE32791)
    ON_COMMAND(ID_REACTIONS_E32792, &MenuDlg::OnReactionsE32792)
    ON_COMMAND(ID_REACTIONS_E32793, &MenuDlg::OnReactionsE32793)
    ON_COMMAND(ID_REACTIONS_HE, &MenuDlg::OnReactionsHe)
    ON_COMMAND(ID_REACTIONS_HE32795, &MenuDlg::OnReactionsHe32795)
    ON_COMMAND(ID_REACTIONS_HE32796, &MenuDlg::OnReactionsHe32796)
    ON_COMMAND(ID_REACTIONS_HE32797, &MenuDlg::OnReactionsHe32797)
    ON_COMMAND(ID_REACTIONS_HE32798, &MenuDlg::OnReactionsHe32798)
    ON_COMMAND(ID_REACTIONS_HE32799, &MenuDlg::OnReactionsHe32799)
    ON_COMMAND(ID_REACTIONS_HE32800, &MenuDlg::OnReactionsHe32800)
    ON_COMMAND(ID_REACTIONS_HE32801, &MenuDlg::OnReactionsHe32801)
    ON_COMMAND(ID_REACTIONS_HE2, &MenuDlg::OnReactionsHe2)
    ON_COMMAND(ID_REACTIONS_HE3, &MenuDlg::OnReactionsHe3)
    ON_COMMAND(ID_HE_HE, &MenuDlg::OnHeHe)
    ON_COMMAND(ID_HE_HE32810, &MenuDlg::OnHeHe32810)
    ON_COMMAND(ID_HE_HE32811, &MenuDlg::OnHeHe32811)
    ON_COMMAND(ID_HE_HE32812, &MenuDlg::OnHeHe32812)
    ON_COMMAND(ID_HE_HE32813, &MenuDlg::OnHeHe32813)
    ON_COMMAND(ID_HE_HE32814, &MenuDlg::OnHeHe32814)
    ON_COMMAND(ID_HE_HE32815, &MenuDlg::OnHeHe32815)
    ON_COMMAND(ID_HE_HE32816, &MenuDlg::OnHeHe32816)
    ON_COMMAND(ID_HE_HE32817, &MenuDlg::OnHeHe32817)
    ON_COMMAND(ID_HE_HE32818, &MenuDlg::OnHeHe32818)
    ON_COMMAND(ID_HE_HE32819, &MenuDlg::OnHeHe32819)
    ON_COMMAND(ID_HE_HE32820, &MenuDlg::OnHeHe32820)
    ON_COMMAND(ID_HE_HE32821, &MenuDlg::OnHeHe32821)
    ON_COMMAND(ID_HE_HE32822, &MenuDlg::OnHeHe32822)
```

```
    ON_COMMAND(ID_HE_HE32823, &MenuDlg::OnHeHe32823)
    ON_COMMAND(ID_HE_HE32824, &MenuDlg::OnHeHe32824)
    ON_COMMAND(ID_HE_HE32825, &MenuDlg::OnHeHe32825)
    ON_COMMAND(ID_HE_HE32826, &MenuDlg::OnHeHe32826)
    ON_COMMAND(ID_HE2_HE2, &MenuDlg::OnHe2He2)
    ON_COMMAND(ID_HE2_HE3, &MenuDlg::OnHe2He3)
    ON_COMMAND(ID_HE2_HE4, &MenuDlg::OnHe2He4)
    ON_COMMAND(ID_HE2_HE5, &MenuDlg::OnHe2He5)
    ON_COMMAND(ID_HE2_HE6, &MenuDlg::OnHe2He6)
    ON_COMMAND(ID_HE2_HE7, &MenuDlg::OnHe2He7)
    ON_COMMAND(ID_HE2_HE8, &MenuDlg::OnHe2He8)
    ON_COMMAND(ID_HE2_HE9, &MenuDlg::OnHe2He9)
    ON_COMMAND(ID_HE2_HE10, &MenuDlg::OnHe2He10)
    ON_COMMAND(ID_E_EDENSITY, &MenuDlg::OnEEdensity)
    ON_COMMAND(ID_E_EDIFFUSIONCOEF, &MenuDlg::OnEEdiffusioncoef)
    ON_COMMAND(ID_E_EMOBILITY, &MenuDlg::OnEEmobility)
    ON_COMMAND(ID_E_ESOURCE, &MenuDlg::OnEEsource)
    ON_COMMAND(ID_E_ELINEARISEDSOURCE, &MenuDlg::OnEElinearisedsource)
        ON_COMMAND(ID_E_EFLUXX, &MenuDlg::OnEEfluxx)
        ON_COMMAND(ID_E_EFLUXY, &MenuDlg::OnEEfluxy)
        ON_COMMAND(ID_E_EPOWERDISSIPATION, &MenuDlg::OnEEpowerdissipation)
        ON_COMMAND(ID_E_EMOBILITYTABLE, &MenuDlg::OnEEmobilitytable)
        ON_COMMAND(ID_E_E, &MenuDlg::OnEE)
        ON_COMMAND(ID_E_E32846, &MenuDlg::OnEE32846)
        ON_COMMAND(ID_E_E32847, &MenuDlg::OnEE32847)
        ON_COMMAND(ID_E_E32848, &MenuDlg::OnEE32848)
        ON_COMMAND(ID_E_E32849, &MenuDlg::OnEE32849)
        ON_COMMAND(ID_E_E32850, &MenuDlg::OnEE32850)
        ON_COMMAND(ID_E_E32851, &MenuDlg::OnEE32851)
        ON_COMMAND(ID_E_E32852, &MenuDlg::OnEE32852)
        ON_COMMAND(ID_E_E32853, &MenuDlg::OnEE32853)
        ON_COMMAND(ID_LINECHART_Q1, &MenuDlg::OnLinechartQ1)
        ON_COMMAND(ID_LINECHART_Q2, &MenuDlg::OnLinechartQ2)
        ON_COMMAND(ID_LINECHART_Q4, &MenuDlg::OnLinechartQ4)
        ON_COMMAND(ID_LINECHART_P0, &MenuDlg::OnLinechartP0)
        ON_EN_CHANGE(IDC_EDIT_SHOWTXT, &MenuDlg::OnEnChangeEditShowtxt)
        END_MESSAGE_MAP()

// MenuDlg


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

```
    m_diaplay = new CMyOpenGLView1();
    CRect  rect = new CRect(30,30,700,600);
    //GetDlgItem(IDC_STATIC_GLTu)->GetWindowRect(&rect);
    //ScreenToClient(&rect);
    m_diaplay->Create(NULL, NULL, WS_VISIBLE, rect, this, 0);

    CMenu* m_menu = new CMenu();
    m_menu->LoadMenu(IDR_MENU_PAGE2);
    this->SetMenu(m_menu);
    //

    //
    char pFileName[MAX_PATH];
    int nPos = GetCurrentDirectory(MAX_PATH, pFileName);

    CString csFullPath(pFileName);

    int col, row;
    m_FileName =
CSVRead::ReadCsvFile("NameManagement\\DataName.csv" ,',',&row,&col);



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

}

////------------1
void MenuDlg::OnElectronMeanenergy()
{

    ShowFile(0);
}


void MenuDlg::OnElectronMeanenergydensity()
{
    ShowFile(1);
}


void MenuDlg::OnElectronElasticenergylosses()
```

```
{
    ShowFile(2);
}


void MenuDlg::OnElectronElasticenergylinearisedlosses()
{
    ShowFile(3);
}


void MenuDlg::OnElectronMea()
{
    ShowFile(4);
}


void MenuDlg::OnElectronMeanenergylinearisedsource()
{
    ShowFile(5);
}


void MenuDlg::OnElectronMeanenergyfluxdensity()
{
    ShowFile(6);
}


void MenuDlg::OnElectronMeanenergyfluxdensity32782()
{
    ShowFile(7);
}


void MenuDlg::OnElectronReducedfield()
{
    ShowFile(8);
}


void MenuDlg::OnElectronReducedfield32784()
{
```

```
    ShowFile(9);
}


void MenuDlg::OnElectronEnergytable()
{
    ShowFile(10);
}


void MenuDlg::OnElectronElasticlosstable()
{
    ShowFile(11);
}
void MenuDlg::OnGastemperatureGastemperature()
{
    ShowFile(12);
}


void MenuDlg::OnReactionsE()
{
    ShowFile(13);
}


void MenuDlg::OnReactionsE32789()
{
    ShowFile(14);
}


void MenuDlg::OnReactionsE32790()
{
    ShowFile(15);
}


void MenuDlg::OnReactionsE32791()
{
    ShowFile(16);
}
```

```
void MenuDlg::OnReactionsE32792()
{
    ShowFile(17);
}


void MenuDlg::OnReactionsE32793()
{
    ShowFile(18);
}


void MenuDlg::OnReactionsHe()
{
    ShowFile(19);
}


void MenuDlg::OnReactionsHe32795()
{
    ShowFile(20);
}


void MenuDlg::OnReactionsHe32796()
{
    ShowFile(21);
}


void MenuDlg::OnReactionsHe32797()
{
    ShowFile(22);
}


void MenuDlg::OnReactionsHe32798()
{
    ShowFile(23);
}
```

```
void MenuDlg::OnReactionsHe32799()
{
    ShowFile(24);
}


void MenuDlg::OnReactionsHe32800()
{
    ShowFile(25);
}


void MenuDlg::OnReactionsHe32801()
{
    ShowFile(26);
}


void MenuDlg::OnReactionsHe2()
{
    ShowFile(27);
}


void MenuDlg::OnReactionsHe3()
{
    ShowFile(28);
}


void MenuDlg::OnHeHe()
{
    ShowFile(29);
}


void MenuDlg::OnHeHe32810()
{
    ShowFile(30);
}


void MenuDlg::OnHeHe32811()
```

```
{
    ShowFile(31);
}


void MenuDlg::OnHeHe32812()
{
    ShowFile(32);
}


void MenuDlg::OnHeHe32813()
{
    ShowFile(33);
}


void MenuDlg::OnHeHe32814()
{
    ShowFile(34);
}


void MenuDlg::OnHeHe32815()
{
    ShowFile(35);
}


void MenuDlg::OnHeHe32816()
{
    ShowFile(36);
}


void MenuDlg::OnHeHe32817()
{
    ShowFile(37);
}


void MenuDlg::OnHeHe32818()
{
```

```
    ShowFile(38);
}


void MenuDlg::OnHeHe32819()
{
    ShowFile(39);
}


void MenuDlg::OnHeHe32820()
{
    ShowFile(40);
}


void MenuDlg::OnHeHe32821()
{
    ShowFile(41);
}


void MenuDlg::OnHeHe32822()
{
    ShowFile(42);
}


void MenuDlg::OnHeHe32823()
{
    ShowFile(43);
}


void MenuDlg::OnHeHe32824()
{
    ShowFile(44);
}


void MenuDlg::OnHeHe32825()
{
    ShowFile(45);
```

```
}


void MenuDlg::OnHeHe32826()
{
    ShowFile(46);
}


void MenuDlg::OnHe2He2()
{
    ShowFile(47);
}


void MenuDlg::OnHe2He3()
{
    ShowFile(48);
}


void MenuDlg::OnHe2He4()
{
    ShowFile(49);
}


void MenuDlg::OnHe2He5()
{
    ShowFile(50);
}


void MenuDlg::OnHe2He6()
{
    ShowFile(51);
}


void MenuDlg::OnHe2He7()
{
    ShowFile(52);
}
```

```
void MenuDlg::OnHe2He8()
{
    ShowFile(53);
}


void MenuDlg::OnHe2He9()
{
    ShowFile(54);
}


void MenuDlg::OnHe2He10()
{
    ShowFile(55);
}


void MenuDlg::OnEEdensity()
{
    ShowFile(56);
}


void MenuDlg::OnEEdiffusioncoef()
{
    ShowFile(57);
}


void MenuDlg::OnEEmobility()
{
    ShowFile(58);
}


void MenuDlg::OnEEsource()
{
    ShowFile(59);
}
```

```
void MenuDlg::OnEElinearisedsource()
{
    ShowFile(60);
}


void MenuDlg::OnEEfluxx()
{
    ShowFile(61);
}


void MenuDlg::OnEEfluxy()
{
    ShowFile(62);
}


void MenuDlg::OnEEpowerdissipation()
{
    ShowFile(63);
}


void MenuDlg::OnEEmobilitytable()
{
    ShowFile(64);
}


void MenuDlg::OnEE()
{
    ShowFile(65);
}


void MenuDlg::OnEE32846()
{
    ShowFile(66);
}
```

```
void MenuDlg::OnEE32847()
{
    ShowFile(67);
}


void MenuDlg::OnEE32848()
{
    ShowFile(68);
}


void MenuDlg::OnEE32849()
{
    ShowFile(69);
}


void MenuDlg::OnEE32850()
{
    ShowFile(70);
}


void MenuDlg::OnEE32851()
{
    ShowFile(71);
}


void MenuDlg::OnEE32852()
{
    ShowFile(72);
}


void MenuDlg::OnEE32853()
{
    ShowFile(73);
}

//-------------------------------------2----------------------------
void MenuDlg::OnEmregionPotential()
```

```
{
    ShowFile(74);
}


void MenuDlg::OnEmregionCurrentdensity()
{
    ShowFile(75);
}


void MenuDlg::OnEmregionEx()
{
    ShowFile(76);
}


void MenuDlg::OnEmregionEy()
{
    ShowFile(77);
}


void MenuDlg::OnEmregionPowerdissipation()
{
    ShowFile(78);
}


void MenuDlg::OnEmregionV()
{
    ShowFile(79);
}


void MenuDlg::OnEmregionVolumechargedensity()
{
    ShowFile(80);
}


void MenuDlg::OnEmregionVolumechargedensity32902()
{
```

```
    ShowFile(81);
}

//----------------------------3-----------------------------------------
void MenuDlg::OnGeneralConfiguration()
{
    ShowFile(82);
}

//----------------------------4-----------------------------------------
void MenuDlg::OnGeometryConfiguration()
{
    ShowFile(83);
}


void MenuDlg::OnGeometryCvvolumes()
{
    ShowFile(84);
}


void MenuDlg::OnGeometryCvareas()
{
    ShowFile(85);
}


void MenuDlg::OnGeometryCvareas32871()
{
    ShowFile(86);
}
//----------------------------5-----------------------------------------

//void MenuDlg::OnLinechartDt()
//{
//  //AfxMessageBox("hello");
//  ShowLine(1);
//}


void MenuDlg::OnLinechartEr()
{
```

```
    ShowLine(2);
}


void MenuDlg::OnLinechartEpsilon()
{
    ShowLine(3);
}


void MenuDlg::OnLinechartN0()
{
    ShowLine(4);
}


void MenuDlg::OnLinechartN1()
{
    ShowLine(5);
}


void MenuDlg::OnLinechartN2()
{
    ShowLine(6);
}


void MenuDlg::OnLinechartN3()
{
    ShowLine(7);
}


void MenuDlg::OnLinechartN4()
{
    ShowLine(8);
}


void MenuDlg::OnLinechartN5()
{
    ShowLine(9);
```

```
}


void MenuDlg::OnLinechartElasloss()
{
    ShowLine(10);
}


void MenuDlg::OnLinechartR0()
{
    ShowLine(11);
}


void MenuDlg::OnLinechartR1()
{
    ShowLine(12);
}


void MenuDlg::OnLinechartR2()
{
    ShowLine(13);
}


void MenuDlg::OnLinechartR3()
{
    ShowLine(14);
}


void MenuDlg::OnLinechartR4()
{
    ShowLine(15);
}

void MenuDlg::OnLinechartR5()
{
    ShowLine(16);
}
```

```
void MenuDlg::OnLinechartR6()
{
    ShowLine(17);
}


void MenuDlg::OnLinechartR7()
{
    ShowLine(18);
}


void MenuDlg::OnLinechartV1()
{
    ShowLine(19);
}


void MenuDlg::OnLinechartV2()
{
    ShowLine(20);
}


void MenuDlg::OnLinechartV4()
{
    ShowLine(21);
}


void MenuDlg::OnLinechartI1()
{
    ShowLine(22);
}


void MenuDlg::OnLinechartIc1()
{
    ShowLine(23);
}
```

```
void MenuDlg::OnLinechartI2()
{
     ShowLine(24);
}


void MenuDlg::OnLinechartIc2()
{
    ShowLine(25);
}


void MenuDlg::OnLinechartI4()
{
    ShowLine(26);
}


void MenuDlg::OnLinechartIc4()
{
    ShowLine(27);

}

void MenuDlg::OnLinechartQ1()
{
    ShowLine(28);
}


void MenuDlg::OnLinechartQ2()
{
    ShowLine(29);
}


void MenuDlg::OnLinechartQ4()
{
    ShowLine(30);
}


void MenuDlg::OnLinechartP0()
```

```
{
    ShowLine(31);
}

//--------------------------------------------------------------------------
---------------------------------
CString  MenuDlg::GetExtensionOfFile(CString filePath)//return the type of file
based on suffix
{
    int dotPos = filePath.ReverseFind('.');
    return (filePath.Right(filePath.GetLength() - dotPos-1));
}
CString  MenuDlg::readInfoText(CString filepath)//Read text file based on name
{

    if (!PathFileExists(filepath ))
        return "";


    CFile   file(filepath, CFile::modeRead);//open text file only for read
    char   *pBuf;
    int   iLen = file.GetLength(); //get the length of data
    pBuf = new   char[iLen + 1];
    file.Read(pBuf, iLen);
    pBuf[iLen] = 0;//add 0 at the end
    file.Close();

    CString A;
    A.Format("%s", pBuf);

    free(pBuf);
    m_EditShowTxt = A;
    UpdateData(FALSE);
    return  A;
}

CString  ** MenuDlg::ReadCsvFile(CString filepath)//obtain the data name from
the csv file
{

    CStringArray * Array = NULL;

    if (!PathFileExists(filepath))
        return NULL;
```

```
    CSVRead read(filepath);

    int Rowcount = read.FileRowCount();
    Array = read.CSVInf(' ');
    int Columcount = Array->GetSize();

    CString **p = new CString *[Rowcount];
    for (int i = 0; i < Rowcount; i++)
        p[i] = new CString[Columcount];
    for (int i = 0; i < Columcount; i++)
        p[0][i] = Array->GetAt(i);


    for (int i = 1; i<Rowcount; i++)
    {
        Array = read.CSVInf(' ');
        for (int j = 0; j < Columcount; j++)
            p[i][j] = Array->GetAt(j);

    }

    return p;

}
void   MenuDlg::ShowFile(int id)
{

    if (!PathFileExists(m_OpenFileFolder + "\\" + m_FileName[id][0])||
m_FileName[id][0]=="")// if not find the name of related text file
        m_EditShowTxt = "No Data Available";
    else
    {
        m_EditShowTxt = readInfoText(m_OpenFileFolder + "\\" +
m_FileName[id][0]);
        if(id != 82)
        m_diaplay->Readtxt(m_OpenFileFolder + "\\" + m_FileName[id][0]);
    }
    UpdateData(FALSE);
```

```
}
//loading image
void MenuDlg::DrawPicToHDC(char * pathfile, UINT ID)
{

    IplImage *image = NULL;
    if (image) cvReleaseImage(&image);

    image = cvLoadImage(pathfile, 1);

    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC = pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
    cimg.CopyOf(image);
    cimg.DrawToHDC(hDC, &rect);
    ReleaseDC(pDC);
}
void DialogName(DialogGl * m_dialoggl, int biaohao)
{
    switch (biaohao)
    {
    case 1:
        m_dialoggl->m_glName = "dt";
        m_dialoggl->m_name = "S";
        break;
    case 2:
        m_dialoggl->m_glName = "Er";
        m_dialoggl->m_name = "Vm-1*Pa-1";
        break;
    case 3:
        m_dialoggl->m_glName = "Epsilon";
        m_dialoggl->m_name = "J";
        break;
    case 4:
        m_dialoggl->m_glName = "no";
        m_dialoggl->m_name = "J ";
        break;
    case 5:
        m_dialoggl->m_glName = "n1";
        m_dialoggl->m_name = " J";
        break;
```

```
case 6:
    m_dialoggl->m_glName = "n2";
    m_dialoggl->m_name = "J*m-3";
    break;
case 7:
    m_dialoggl->m_glName = "n3";
    m_dialoggl->m_name = "J*m-3";
    break;
case 8:
    m_dialoggl->m_glName = "n4";
    m_dialoggl->m_name = "J*m-3";
    break;
case 9:
    m_dialoggl->m_glName = "n5";
    m_dialoggl->m_name = "J*m-3";
    break;
case 10:
    m_dialoggl->m_glName = "ElasLoss";
    m_dialoggl->m_name = "J";
    break;
case 11:
    m_dialoggl->m_glName = "r0";
    m_dialoggl->m_name = "J";
    break;
case 12:
    m_dialoggl->m_glName = "r1";
    m_dialoggl->m_name = "J";
    break;
case 13:
    m_dialoggl->m_glName = "r2";
    m_dialoggl->m_name = "J";
    break;
case 14:
    m_dialoggl->m_glName = "r3";
    m_dialoggl->m_name = "m-3*s-1";
    break;
case 15:
    m_dialoggl->m_glName = "r5";
    m_dialoggl->m_name = "m-3*s-1";
    break;
case 16:
    m_dialoggl->m_glName = "r6";
    m_dialoggl->m_name = "m-3*s-1";
```

```
        break;
    case 17:
        m_dialoggl->m_glName = "r7";
        m_dialoggl->m_name = "m-3*s-1";
        break;
    case 18:
        m_dialoggl->m_glName = "V1";
        m_dialoggl->m_name = "m-3*s-1";
        break;
    case 19:
        m_dialoggl->m_glName = "V2";
        m_dialoggl->m_name = "V";
        break;
    case 20:
        m_dialoggl->m_glName = "V4";
        m_dialoggl->m_name = "V";
        break;
    case 21:
        m_dialoggl->m_glName = "I1";
        m_dialoggl->m_name = "V";
        break;
    case 22:
        m_dialoggl->m_glName = "Ic1";
        m_dialoggl->m_name = "V";
        break;
    case 23:
        m_dialoggl->m_glName = "I2";
        m_dialoggl->m_name = "A";
        break;
    case 24:
        m_dialoggl->m_glName = "Ic2";
        m_dialoggl->m_name = "A";
        break;
    case 25:
        m_dialoggl->m_glName = "I4";
        m_dialoggl->m_name = "A";
        break;
    case 26:
        m_dialoggl->m_glName = "Ic4";
        m_dialoggl->m_name = "A";
        break;
    case 27:
        m_dialoggl->m_glName = "Q1";
```

```
            m_dialoggl->m_name = "A";
            break;
        case 28:
            m_dialoggl->m_glName = "Q2";
            m_dialoggl->m_name = "C";
            break;
        case 29:
            m_dialoggl->m_glName = "Q4";
            m_dialoggl->m_name = "C";
            break;
        case 30:
            m_dialoggl->m_glName = "P0";
            m_dialoggl->m_name = "W*m-3";
            break;
        case 31:
            break;
        default:
            break;
        }



}
void MenuDlg::ShowLine(int biaohao)
{
    DialogGl * m_dialoggl = new DialogGl();;
    DialogName(m_dialoggl, biaohao);

    m_dialoggl->Create(IDD_DIALOG_GL);
    m_dialoggl->ShowWindow(true);

    m_dialoggl->ReadCsvFile(m_OpenFileFolder + "\\history.out", ' ');
    m_dialoggl->ShowLine(biaohao);


}

void MenuDlg::OnEnChangeEditShowtxt()
{

}
```

MyOpenGLView.cpp

```
#include "StdAfx.h"
#include "MyOpenGLView.h"
#include <math.h>


#define TIMERID 8
 float scaleview;




CMyOpenGLView::CMyOpenGLView(void)
{
        //scaleview=0.015;
        //scaleview=0.1;
        lbuttondown=FALSE;
        drawrobottype=0;
        m_dis1=false;
        m_dis2=false;
        biaohao_show = -1;
        data_row = 0;
        data_col = 0;

}


CMyOpenGLView::~CMyOpenGLView(void)
{
}


BEGIN_MESSAGE_MAP(CMyOpenGLView, CWnd)
    //{{AFX_MSG_MAP(CMyOpenGLView)
    ON_WM_CREATE()
    ON_WM_PAINT()
    ON_WM_SIZE()
    ON_WM_LBUTTONDOWN()
    ON_WM_LBUTTONUP()
    ON_WM_MOUSEMOVE()
```

```
    ON_WM_LBUTTONDBLCLK()
    ON_WM_MOUSEWHEEL()
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
    ON_WM_DESTROY()

END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CMyOpenGLView message handlers

int CMyOpenGLView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    // TODO: Add your specialized creation code here
    hdc=::GetDC(m_hWnd);
    SetupPixelFormat(hdc);
    CPaintDC dc(this);
    //hglrc= wglCreateContext(hdc);
    hglrc= wglCreateContext(hdc);
    wglMakeCurrent(hdc,hglrc);
    ::glClearDepth(1.0f);
    ::glEnable(GL_DEPTH_TEST);
    //::glClearColor(1.0f,0.0f,1.0f,1.0f);

    //delete(hdc);
    return 0;
}

void CMyOpenGLView::OnPaint() //Darw curve
{
    CPaintDC dc(this); // device context for painting
    wglMakeCurrent(hdc,hglrc);
    ::glClearColor(0.0f,0.0f,0.0f,1.0f);
    // TODO: Add your message handler code here
    ::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );//Clear Buffer

    glPushMatrix();//save the matrix
    glTranslatef(-gl_width / 2+1, -gl_heigth/2+1, 0);//location transf
    GLfloat curSizeLine = 1;
```

```
    glPointSize(curSizeLine);
    if (biaohao_show <= 1)//The first colume is time
        return ;
    glBegin(GL_POINTS);
    glColor3f(1.0f, 0.0f, 0.0f);

    for (int i = 1; i < data_row; i++)
    {

        glVertex2f(atof(data[i][0]) / (maxx - minx)*gl_width,
atof(data[i][biaohao_show]) / (maxy - miny)*gl_heigth);//draw all points

    }

    glEnd();
    glPopMatrix();
    ::glFinish();       // End draw
    SwapBuffers( hdc);
    wglMakeCurrent(hdc,NULL);
    Sleep(1);
    // Do not call CWnd::OnPaint() for painting messages
}

void CMyOpenGLView::drawrobot(int type)
{

    if(type==1)
    {
        DrawTu1();




    }
    else
    {


   glEnd();

    }

}
```

```
void CMyOpenGLView::DrawTu1()
{
      // glTranslated(-1.0,1.0 ,0.0 );


}
void CMyOpenGLView::DrawTu2()
{
}

BOOL CMyOpenGLView::SetupPixelFormat(HDC hdc)
{
    PIXELFORMATDESCRIPTOR pfd=
    {
        sizeof(PIXELFORMATDESCRIPTOR),
            1,
            PFD_DRAW_TO_WINDOW | //The buffer can draw to a window or device
surface
            PFD_SUPPORT_OPENGL |//The buffer supports OpenGL drawing.
            PFD_DOUBLEBUFFER,//The buffer is double-buffered. This flag and
PFD_SUPPORT_GDI are mutually exclusive in the current generic implementation.
            PFD_TYPE_RGBA,//RGBA pixels. Each pixel has four components in this
order: red, green, blue, and alpha.
            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( hdc, &pfd))  )
    {

        return FALSE;
```

```
    }

    if( FALSE == ::SetPixelFormat( hdc,
        pixelformat, &pfd) )
    {

        return FALSE;
    }

    return TRUE;

}

void CMyOpenGLView::OnSize(UINT nType, int cx, int cy)
{
    CWnd::OnSize(nType, cx, cy);

    // TODO: Add your message handler code here
    GLdouble aspect_ratio;

    if( 0 >= cx || 0 >= cy )
        return;
    gl_width = cx;
    gl_heigth = cy;
    SetupViewPort( cx, cy );//setting size of picture

    // compute the aspect ratio
    aspect_ratio = (GLdouble)cx / (GLdouble)cy;//the ratio of length and width

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-cx / 2, cx / 2, -cy / 2, cy / 2);


    ::glMatrixMode(GL_MODELVIEW);


    ::glLoadIdentity();//reset



}
```

```
BOOL CMyOpenGLView::SetupViewPort(int cx, int cy)
{
    glViewport(0, 0, cx, cy);//redraw based on the change of window

    return TRUE;
}

BOOL CMyOpenGLView::SetupViewingFrustum(GLdouble aspect_ratio)
{
    gluPerspective( 40.0f, aspect_ratio, 0.1f, 20.0f );
    return TRUE;
}

BOOL CMyOpenGLView::SetupViewingTransform()
{
    int i;

    GLfloat fovy = 30.0;
    GLfloat eye[3];
    GLfloat center[3] = { 0.0f, 0.0f, 1.0f };
    GLfloat eye_dir[3];
    GLfloat up[3];
    GLfloat norm, dist;

    eye[0] = 0.0;
    eye[1] = 5.0;
    eye[2] = 2.0;

    for(i=0; i<3; i++)
        eye_dir[i] = center[i] - eye[i];
    dist = (GLfloat)sqrt( eye_dir[0]*eye_dir[0] + eye_dir[1]*eye_dir[1] +
eye_dir[2]*eye_dir[2] );
    for(i=0; i<3; i++)
        eye_dir[i] /= (GLfloat)dist;



    up[0] = -eye_dir[0] * eye_dir[2];
    up[1] = -eye_dir[1] * eye_dir[2];
    up[2] = eye_dir[0] * eye_dir[0] + eye_dir[1] * eye_dir[1];
    norm = up[1]*up[1] + up[1]*up[1] + up[2]*up[2];
    norm = (GLfloat)sqrt(norm);
    for(i=0; i<3; i++)
```

```
        up[i] /= norm;
    gluLookAt( eye[0], eye[1],    eye[2],
        center[0], center[1], center[2],
        up[0],    up[1],     up[2]);
    return TRUE;
}

BOOL CMyOpenGLView::SetupLighting()
{


    GLfloat model_ambient[] = { 2.0f, 2.0f, 2.0f, 1.0f };//default value
    GLfloat light_position0[]= { 1.0f, 0.0f, 5.0f, 0.0f };
    GLfloat light_color0[] = { 1.0f, 1.0f, 1.0f, 1.0f };


    glLightModelfv( GL_LIGHT_MODEL_AMBIENT, model_ambient );


    glLightfv( GL_LIGHT0, GL_POSITION, light_position0 );


    glLightfv( GL_LIGHT0, GL_DIFFUSE, light_color0 );


    glEnable( GL_LIGHTING );
    glEnable( GL_LIGHT0 );

    return TRUE;
}

void CMyOpenGLView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    /*lbuttondown=TRUE;
    pointcurrent=point;
    SetFocus();*/
    CWnd::OnLButtonDown(nFlags, point);
}

void CMyOpenGLView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
```

```
    /*lbuttondown=FALSE;*/

    CWnd::OnLButtonUp(nFlags, point);
}

void CMyOpenGLView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default


    CWnd::OnMouseMove(nFlags, point);
}
void CMyOpenGLView::getdc()
{
    ::GetDC(m_hWnd);
}

void CMyOpenGLView::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    CWnd::OnLButtonDblClk(nFlags, point);
}

void CMyOpenGLView::OnPaint2(int type)
{
    ::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );//clear buffer

        ::glPushMatrix();


        ::glPopMatrix();



    ::glFinish();

    SwapBuffers( hdc);


}

BOOL CMyOpenGLView::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
```

```
{
    // TODO: Add your message handler code here and/or call default


    return CWnd::OnMouseWheel(nFlags, zDelta, pt);
}

void CMyOpenGLView::OnDestroy()
{

    CWnd::OnDestroy();

}

void CMyOpenGLView::ReadFile(CString pathfile, char divided)//Read file and get
data
{
    data_col = 0;
    data_row = 0;
    data = CSVRead::ReadCsvFile(pathfile, divided,&data_row,&data_col);

}
float*  CMyOpenGLView::ShowLine(int biaohao,float *kedu)
{
    biaohao_show = biaohao;
    minx = -1; miny = -1; maxx = -1; maxy = -1;

    if (data == NULL)
        return kedu;

        for(int i = 1; i < data_row; i++)
        {
            if (minx == -1)
            minx = atof(data[i][0]);
            if (maxx == -1)
            maxx = atof(data[i][0]);
            if (miny == -1)
            miny = atof(data[i][biaohao]);
            if (maxy == -1)
            maxy = atof(data[i][biaohao]);

            CString a = data[i][0];
            CString b = data[i][biaohao];
```

```
            minx = atof(data[i][0]) < minx ? atof(data[i][0]) : minx;
            maxx = atof(data[i][0]) > maxx ? atof(data[i][0]) : maxx;

            miny = atof(data[i][biaohao]) < miny ? atof(data[i][biaohao]) :
miny;
            maxy = atof(data[i][biaohao]) > maxy ? atof(data[i][biaohao]) :
maxy;

        }

        kedu[0] = minx;
        kedu[1] = maxx;
        kedu[2] = miny;
        kedu[3] = maxy;

        SetTimer(TIMERID, 10, 0);
        return kedu;
}
void CMyOpenGLView::OnTimer(UINT_PTR nIDEvent)
{
    OnPaint();

}
```

MyOpenGLView1.cpp

```
#define _CRT_SECURE_NO_WARNINGS 1
#include "StdAfx.h"
#include "MyOpenGLView1.h"
#include <math.h>
#include <sstream>
#include <vector>

#define TIMERID 8
// float scaleview;
#define maxnum 150
std::vector<HangVert> colorpoints[maxnum];
std::vector<std::string> timestring[maxnum] ;
std::vector<int> timesheight[maxnum];
double min[maxnum] = { 10000000000000000000 };
double max[maxnum] = { -10000000000000000000 };
float x, y = 0;
```

```
GLuint mTexts;
void LoadASCII()//for displaying characs in openGL
{
    mTexts = glGenLists(96);
    HFONT hFont = CreateFontA(24, 0, 0, 0, 300, FALSE, FALSE, FALSE,
ANSI_CHARSET, OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
        DEFAULT_QUALITY, DEFAULT_QUALITY, "Arial");

    SelectObject(GetDC(GetActiveWindow()), hFont);
    wglUseFontBitmaps(GetDC(GetActiveWindow()), 32, 96, mTexts);
}
char *LoadFileContent(const char *path)
{
    FILE *pFile = fopen(path, "rb");
    if (pFile)
    {
        fseek(pFile, 0, SEEK_END);
        int nLen = ftell(pFile);
        char *buffer = new char[nLen + 1];
        if (nLen != 0)
        {
            rewind(pFile);
            fread(buffer, nLen, 1, pFile);
            buffer[nLen] = '\0';
        }
        else
        {
            //printf("load file %s fail\n", path);
        }
        fclose(pFile);
        return buffer;
    }
    else
    {
        //printf("open file %s fail\n", path);
    }
    fclose(pFile);
    return  nullptr;

}
void CMyOpenGLView1::Readtxt(const char* filePath)
{
    int num = -1;
    for (int  i = 0; i < maxnum; i++)
```

```
{
    colorpoints[i].clear();
    min[i] = 1000000000000000000;//The min value of text file
    max[i] = -1000000000000000000;//The max value of text file

    timestring[i].clear();//storage of time
    colorpoints[i].clear();//storage of data in text
    timesheight[i].clear();//the space in each time
}

//char*fileContent = LoadFileContent(filePath);
 FILE *fp;
 fp = fopen(filePath, "r");
//std::stringstream ssObjFile(fileContent);
char szOneLine[1000];
std::string temp;
int heigth = 0;
//while (!ssObjFile.eof())
while (!feof(fp))
{
    temp.clear();

    memset(szOneLine, 0, 1000);//Read one row
    fgets(szOneLine, 1000, fp);

    //ssObjFile.getline(szOneLine, 256);
    if (strlen(szOneLine) > 0)
    {
        HangVert vi;
        std::stringstream ssOneLine(szOneLine);

        if (szOneLine[0] != '#')//if not "#" it's the data
        {
            for (int i = 0; i < 60; i++)//obtain the data in each row
            {
                ssOneLine >> vi.colorpoint[i];
                if (vi.colorpoint[i] != 0 && vi.colorpoint[i] != 1)
                {
                    if (vi.colorpoint[i] > max[num])
                        max[num] = vi.colorpoint[i];
                    if(vi.colorpoint[i] < min[num])
                        min[num] = vi.colorpoint[i];
                }
```

```
                }
                colorpoints[num].push_back(vi);
            }
            else if(szOneLine[0] == '#'&&szOneLine[2] == 't')//if is "#" it
mean it is the start of a series data of time
            {
                num++;
                //give a different color to different two time images
                for (int i = 0; i < 60; i++)
                    vi.colorpoint[i] = 1010;
                colorpoints[num].push_back(vi);
                for (int i = 0; i < 60; i++)
                    vi.colorpoint[i] = 1010;
                colorpoints[num].push_back(vi);

                //save the charater of time
                ssOneLine >> temp;
                ssOneLine >> temp;
                ssOneLine >> temp;
                ssOneLine >> temp;
                timestring[num].push_back(temp);
                timesheight[num].push_back(heigth);

            }
            else
            {
                int a = 0;

            }

            heigth++;
        }

    }
    fclose(fp);
    SetTimer(TIMERID, 10, 0);
}
void setcolor(double num)
{
    //Based on different value of data, drawing different colors

    int a = 1;
     if(num<=2)
```

```
    {

        glColor3f(1*a, 0, 1*a);
    }
    else if (num < 3 && num >= 2)
    {

        glColor3f(0, 0, 1 * a);

    }
    else if (num < 4 && num >= 3)
    {

        glColor3f(0, 1*a, 1 * a);

    }
    else if (num < 5 && num >= 4)
    {

        glColor3f(0, 1 * a, 0 );

    }
    else if (num < 6 && num >= 5)
    {

        glColor3f(1*a, 1 * a, 0);

    }
    else if (num < 7 && num >= 6)
    {

        glColor3f(1 * a, 0.64 * a, 0);
    }
    else if (num>= 7)
    {

        glColor3f(1 * a, 0, 0);
    }

}
CMyOpenGLView1::CMyOpenGLView1(void)
{
```

```
        lbuttondown=FALSE;
        drawrobottype=0;

}


CMyOpenGLView1::~CMyOpenGLView1(void)
{
}

BEGIN_MESSAGE_MAP(CMyOpenGLView1, CWnd)
    //{{AFX_MSG_MAP(CMyOpenGLView)
    ON_WM_CREATE()
    ON_WM_PAINT()
    ON_WM_SIZE()
    ON_WM_LBUTTONDOWN()
    ON_WM_LBUTTONUP()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONDBLCLK()
    ON_WM_MOUSEWHEEL()
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
    ON_WM_DESTROY()

    ON_WM_KEYUP()
    ON_WM_KILLFOCUS()
    ON_WM_KEYDOWN()
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CMyOpenGLView message handlers

int CMyOpenGLView1::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    // TODO: Add your specialized creation code here
    hdc=::GetDC(m_hWnd);
    SetupPixelFormat(hdc);
    CPaintDC dc(this);
    //hglrc= wglCreateContext(hdc);
```

```
    hglrc= wglCreateContext(hdc);
    wglMakeCurrent(hdc,hglrc);
    ::glClearDepth(1.0f);
    ::glEnable(GL_DEPTH_TEST);
    //::glClearColor(1.0f,0.0f,1.0f,1.0f);

    //delete(hdc);
    return 0;
}

void CMyOpenGLView1::OnPaint() //opengl drawing
{
    CPaintDC dc(this); // device context for painting
    wglMakeCurrent(hdc, hglrc);
    ::glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    // TODO: Add your message handler code here
    ::glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);//clear buffer


    int height = 0;

    glPushMatrix();
    glTranslated(30, -0 + y, -250);
    glScaled(1, 1, 1);



    for (int j = 0; j < maxnum; j++)
    {
        if (colorpoints[j].size() > 1)
        {
            int mm_height = 0;
            //glPushMatrix(); draw time
            for (int i = 0; i < timestring[j].size(); i++)//timestring.size()
            {


                glColor3f(1, 1, 1);
                glRasterPos3f(-100, -timesheight[j][i] * 4 + 10 * i + y, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(timestring[j][i].c_str()), GL_BYTE,
timestring[j][i].c_str());
```

```
                mm_height = timesheight[j][i] * 4 ;
            }


            {
                float mid = (max[j] - min[j]) / 7.0;
                float x1 = min[j] + mid;
                float x2 = min[j] + 2 * mid;
                float x3 = min[j] + 3 * mid;
                float x4 = min[j] + 4 * mid;
                float x5 = min[j] + 5 * mid;
                float x6 = min[j] + 6 * mid;
                float x7 = min[j] + 7 * mid;

                CString s1;
                s1.Format("Purple  :  %.2e -- %.2e", min, x1);
                CString s2;
                s2.Format("Blue    : %.2e -- %.2e", x1, x2);
                CString s3;
                s3.Format("cyan    : %.2e -- %.2e", x2, x3);
                CString s4;
                s4.Format("Green   : %.2e -- %.2e", x3, x4);
                CString s5;
                s5.Format("Yellow  : %.2e -- %.2e", x4, x5);
                CString s6;
                s6.Format("Orange  : %.2e -- %.2e", x5, x6);
                CString s7;
                s7.Format("Red    : : %.2e -- %.2e", x6, x7);

                float  midx = -200, midy = -mm_height +y ;

                glColor3f(1, 0, 1);
                glRasterPos3f(midx, -15+ midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s1.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s1.GetBuffer()).c_str());

                glColor3f(0, 0, 1);
                glRasterPos3f(midx, -30 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s2.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s2.GetBuffer()).c_str());
```

```
                glColor3f(0, 1, 1);
                glRasterPos3f(midx, -45 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s3.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s3.GetBuffer()).c_str());

                glColor3f(0, 1, 0);
                glRasterPos3f(midx, -60 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s4.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s4.GetBuffer()).c_str());

                glColor3f(1, 1, 0);
                glRasterPos3f(midx, -75 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s5.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s5.GetBuffer()).c_str());

                glColor3f(1, 0.64, 0);
                glRasterPos3f(midx, -90 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s6.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s6.GetBuffer()).c_str());

                glColor3f(1, 0, 0);
                glRasterPos3f(midx, -105 + midy, -250);
                glListBase(mTexts - 32);
                glCallLists(strlen(((std::string)s7.GetBuffer()).c_str()),
GL_BYTE, ((std::string)s7.GetBuffer()).c_str());

            }



            double mid = 7 / (max[j] - min[j]);
            double addnum = 1 - min[j]*mid;
            glPushMatrix();
            glTranslatef(-30, 0, 0);
            glScaled(4, 4, 1);
            glEnable(GL_POINT_SMOOTH);
            glPointSize(6);
            glBegin(GL_POINTS);
```

```
            //Draw the image of data
            for (int i = 0; i < colorpoints[j].size(); i++)
            {


                for (int k = 0; k < 60; k++)
                {


                    glColor3f(0.1, 0.2, 0.3);
                    if (colorpoints[j][i].colorpoint[k] == 1010)
                        glColor3f(0.3, 0.2, 0.1);
                    else if (colorpoints[j][i].colorpoint[k] != 0 &&
colorpoints[j][i].colorpoint[k] != 1)
                        setcolor(colorpoints[j][i].colorpoint[k] * mid +
addnum); //transf the range to number 1~7
                    glVertex2f(0.5*k, -0.5*i- height);

                }


            }
            height += 0.5*colorpoints[j].size();
            glEnd();
            glPopMatrix();
        }
    }

    glPopMatrix();
    ::glFinish();       // end of drawing
    SwapBuffers( hdc);
    wglMakeCurrent(hdc,NULL);
    Sleep(1);
    // Do not call CWnd::OnPaint() for painting messages
}
void CMyOpenGLView1::drawrobot(int type)
{

    if(type==1)
    {
        DrawTu1();
```

```
    }
    else
    {


    glEnd();

    }

}

void CMyOpenGLView1::DrawTu1()
{
      // glTranslated(-1.0,1.0 ,0.0 );


}
void CMyOpenGLView1::DrawTu2()
{
}

BOOL CMyOpenGLView1::SetupPixelFormat(HDC hdc)
{
    PIXELFORMATDESCRIPTOR pfd=
    {
        sizeof(PIXELFORMATDESCRIPTOR),
            1,
            PFD_DRAW_TO_WINDOW | //The buffer can draw to a window or device
surface
            PFD_SUPPORT_OPENGL |//The buffer supports OpenGL drawing.
            PFD_DOUBLEBUFFER,//The buffer is double-buffered. This flag and
PFD_SUPPORT_GDI are mutually exclusive in the current generic implementation.
            PFD_TYPE_RGBA,//RGBA pixels. Each pixel has four components in this
order: red, green, blue, and alpha.
            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( hdc, &pfd))  )
    {

        return FALSE;
    }

    if( FALSE == ::SetPixelFormat( hdc,
        pixelformat, &pfd) )
    {

        return FALSE;
    }

    return TRUE;

}

void CMyOpenGLView1::OnSize(UINT nType, int cx, int cy)
{
    CWnd::OnSize(nType, cx, cy);

    // TODO: Add your message handler code here
    GLdouble aspect_ratio;

    if( 0 >= cx || 0 >= cy )
        return;
    gl_width = cx;
    gl_heigth = cy;
    SetupViewPort( cx, cy );//setting size

    // compute the aspect ratio
    aspect_ratio = (GLdouble)cx / (GLdouble)cy;//The ratio of length and width

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

```
    //gluOrtho2D(-cx / 2, cx / 2, -cy / 2, cy / 2);
    gluPerspective(45.0f, (GLfloat)cx / (GLfloat)cy, 0.1f, 10000.0f);//
Calculate The Aspect Ratio Of The Window


    ::glMatrixMode(GL_MODELVIEW);


    ::glLoadIdentity();//reset



}

BOOL CMyOpenGLView1::SetupViewPort(int cx, int cy)
{
    LoadASCII();
    glViewport(0, 0, cx, cy);//Redraw the dialog  based on the change

    return TRUE;
}

BOOL CMyOpenGLView1::SetupViewingFrustum(GLdouble aspect_ratio)
{
    gluPerspective( 40.0f, aspect_ratio, 0.1f, 20.0f );
    return TRUE;
}

BOOL CMyOpenGLView1::SetupViewingTransform()//openGL setting
{
    int i;

    GLfloat fovy = 30.0;
    GLfloat eye[3];
    GLfloat center[3] = { 0.0f, 0.0f, 1.0f };
    GLfloat eye_dir[3];
    GLfloat up[3];
    GLfloat norm, dist;

    eye[0] = 0.0;
    eye[1] = 5.0;
    eye[2] = 2.0;
```

```
    for(i=0; i<3; i++)
        eye_dir[i] = center[i] - eye[i];
    dist = (GLfloat)sqrt( eye_dir[0]*eye_dir[0] + eye_dir[1]*eye_dir[1] +
eye_dir[2]*eye_dir[2] );
    for(i=0; i<3; i++)
        eye_dir[i] /= (GLfloat)dist;


    up[0] = -eye_dir[0] * eye_dir[2];
    up[1] = -eye_dir[1] * eye_dir[2];
    up[2] = eye_dir[0] * eye_dir[0] + eye_dir[1] * eye_dir[1];
    norm = up[1]*up[1] + up[1]*up[1] + up[2]*up[2];
    norm = (GLfloat)sqrt(norm);
    for(i=0; i<3; i++)
        up[i] /= norm;
    gluLookAt( eye[0], eye[1],    eye[2],
        center[0], center[1], center[2],
        up[0],    up[1],     up[2]);
    return TRUE;
}

BOOL CMyOpenGLView1::SetupLighting()//OpenGL setting
{

    //setting lighting
    GLfloat model_ambient[] = { 2.0f, 2.0f, 2.0f, 1.0f };
    GLfloat light_position0[]= { 1.0f, 0.0f, 5.0f, 0.0f };
    GLfloat light_color0[] = { 1.0f, 1.0f, 1.0f, 1.0f };


    glLightModelfv( GL_LIGHT_MODEL_AMBIENT, model_ambient );


    glLightfv( GL_LIGHT0, GL_POSITION, light_position0 );


    glLightfv( GL_LIGHT0, GL_DIFFUSE, light_color0 );


    glEnable( GL_LIGHTING );
    glEnable( GL_LIGHT0 );

    return TRUE;
```

```
}

void CMyOpenGLView1::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    this->SetFocus();
    CWnd::OnLButtonDown(nFlags, point);
}

void CMyOpenGLView1::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default


    CWnd::OnLButtonUp(nFlags, point);
}

void CMyOpenGLView1::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default


    CWnd::OnMouseMove(nFlags, point);
}
void CMyOpenGLView1::getdc()
{
    ::GetDC(m_hWnd);
}

void CMyOpenGLView1::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    CWnd::OnLButtonDblClk(nFlags, point);
}

void CMyOpenGLView1::OnPaint2(int type)
{
    ::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );//clear buffer

        ::glPushMatrix();// put to the top matrix
```

```
        ::glPopMatrix();//pop the top matrix

    ::glFinish();   //end of draw

    SwapBuffers( hdc);


}

BOOL CMyOpenGLView1::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)
{
    // TODO: Add your message handler code here and/or call default

    y -= 0.5*zDelta; // move speed of mouse wheel

    return CWnd::OnMouseWheel(nFlags, zDelta, pt);
}

void CMyOpenGLView1::OnDestroy()
{

    CWnd::OnDestroy();
    // TODO: 在此处添加消息处理程序代码

}
void CMyOpenGLView1::OnTimer(UINT_PTR nIDEvent)
{
    OnPaint();

}

void CMyOpenGLView1::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    y -= 5; //move speed of keyboard
    CWnd::OnKeyUp(nChar, nRepCnt, nFlags);
}


void CMyOpenGLView1::OnKillFocus(CWnd* pNewWnd)
{
    CWnd::OnKillFocus(pNewWnd);
}
```

```
void CMyOpenGLView1::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    y += 5;//move speed of keyboard
    CWnd::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

MyPropertySheet.cpp

```
// MyPropertySheet.cpp :
//

#include "stdafx.h"
#include "FistPageMain.h"
#include "MyPropertySheet.h"


// MyPropertySheet

IMPLEMENT_DYNAMIC(MyPropertySheet, CPropertySheet)

MyPropertySheet::MyPropertySheet(UINT nIDCaption, CWnd* pParentWnd, UINT
iSelectPage)
    :CPropertySheet(nIDCaption, pParentWnd, iSelectPage)
{

}

MyPropertySheet::MyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd, UINT
iSelectPage)
    :CPropertySheet(pszCaption, pParentWnd, iSelectPage)
{

}

MyPropertySheet::~MyPropertySheet()
{
}


BEGIN_MESSAGE_MAP(MyPropertySheet, CPropertySheet)
END_MESSAGE_MAP()
```

```
BOOL MyPropertySheet::OnInitDialog()
{
    BOOL bResult = CPropertySheet::OnInitDialog();

    GetDlgItem(IDOK)->SetWindowText(_T("OK"));
    GetDlgItem(IDCANCEL)->ShowWindow(SW_HIDE);
    GetDlgItem(IDHELP)->ShowWindow(SW_HIDE);
    GetDlgItem(0x3021)->ShowWindow(SW_HIDE);
    return bResult;
}
```

Page1.cpp

```
// Page1.cpp :
//

#include "stdafx.h"
#include "FistPageMain.h"
#include "Page1.h"
#include "afxdialogex.h"


// CPage1

IMPLEMENT_DYNAMIC(CPage1, CPropertyPage)

CPage1::CPage1()
    : CPropertyPage(IDD_PAGE1)

    , m_editText(_T(""))
{

}

CPage1::~CPage1()
{
}

void CPage1::DoDataExchange(CDataExchange* pDX)
{
```

```
    CPropertyPage::DoDataExchange(pDX);

    DDX_Text(pDX, IDC_EDIT1, m_editText);
}


BEGIN_MESSAGE_MAP(CPage1, CPropertyPage)
    ON_WM_PAINT()
    ON_EN_CHANGE(IDC_EDIT1, &CPage1::OnEnChangeEdit1)
END_MESSAGE_MAP()

//-----------------------------------------
bool CPage1::readInfoText(CString filepath)//read file based on address
{

    if (!PathFileExists(filepath + "\\" + "info.out"))
        return false;


    CFile   file(filepath + "\\" + "info.out", CFile::modeRead);//open text
file for noly read
    char   *pBuf;
    int   iLen = file.GetLength(); //get the length of file
    pBuf = new   char[iLen + 1];
    file.Read(pBuf, iLen);
    pBuf[iLen] = 0;//add 0 to the last new row
    file.Close();

    SetDlgItemText(IDC_EDIT1, pBuf);


    free(pBuf); //release memory

    return  true;
}
void CPage1::DrawPicToHDC(char * pathfile, UINT ID)
{

    IplImage *image = NULL; //The original image
    if (image) cvReleaseImage(&image);

    image = cvLoadImage(pathfile, 1); //display the image
```

```
    CDC *pDC = GetDlgItem(ID)->GetDC();
    HDC hDC = pDC->GetSafeHdc();
    CRect rect;
    GetDlgItem(ID)->GetClientRect(&rect);
    CvvImage cimg;
    cimg.CopyOf(image); // copyy
    cimg.DrawToHDC(hDC, &rect); // put image to widget
    ReleaseDC(pDC);
}
void  CPage1::Initial()
{
    DrawPicToHDC("externResouce\\FirstPicture\\Background.png",
IDC_PICTURE1_BACKGROUND);//background

}

void CPage1::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    Initial();
}


void CPage1::OnEnChangeEdit1()
{

}
```

stdafx.cpp

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

SVRead.cpp

```
#include "StdAfx.h"
#include "SVRead.h"

CSVRead::CSVRead(CString ReadFilePath)
{
    file.Open(ReadFilePath,CFile::typeText|CFile::modeRead);
    file.SeekToBegin();
}

CSVRead::~CSVRead(void)
```

```
{
    file.Close();
}



CStringArray* CSVRead::DevideStr(CString ItemFieldStr , char divider) //Obtain
data from CSV file
{
    CStringArray *SingleItemArray=new CStringArray;
    CStringArray *tempArray=new CStringArray;
    CString SingleItem;
    int dividerPos,dividerSum=0;
    dividerPos=ItemFieldStr.Find(divider,0);
    //counting the number of dividers
    while(dividerPos>=0)
    {
        dividerSum++;
        dividerPos=ItemFieldStr.Find(divider,dividerPos+1);
    }
    for(int i=0;i<=dividerSum;i++)
    {
        AfxExtractSubString(SingleItem,ItemFieldStr,i,divider);
        SingleItemArray->Add(SingleItem);

    }
    SingleItemArray->Add('\0');
    return SingleItemArray;
}

//if not get the right data, delete this data
CStringArray* CSVRead::CSVInf(char divided)
{
    CStringArray* tempString;
    CString str;
    file.ReadString(str);

    if(divided == ' ')
    str = DeleteBlank(str);
    tempString=DevideStr(str, divided);
    return tempString;
}
```

```
// counting the number of rows of csv
int CSVRead::FileRowCount(void)
{
    int countline=0;
    CString sss;
    while(file.ReadString(sss))
        countline++;
    file.SeekToBegin();
    return countline;
}
//transf continuous blank to one
CString CSVRead::DeleteBlank(CString str)
{
    CString reStr = "";
    for (int i = 0; i < str.GetLength()-1; i++)
    {
            if (str[i] == ' '&& str[i + 1] == ' ')
            {

            }
            else
                reStr += str[i];

        if(i == str.GetLength() - 1)
        reStr += str[i];
   }

    return reStr;
}
 CString ** CSVRead::ReadCsvFile(CString filepath, char divided,int *row,int
*col)//Read csv file, get related data and save into an arrary
{
    CStringArray * Array = NULL;

    if (!PathFileExists(filepath))
        return NULL;
    CSVRead read(filepath);

    int Rowcount = read.FileRowCount();
    Array = read.CSVInf(divided);
    int Columcount = Array->GetSize();

    CString **p = new CString *[Rowcount];
```

```
    for (int i = 0; i < Rowcount; i++)
        p[i] = new CString[Columcount];
    for (int i = 0; i < Columcount; i++)
        p[0][i] = Array->GetAt(i);

    *row = Rowcount;
    *col = Columcount;
    for (int i = 1; i<Rowcount ; i++)
    {
        Array = read.CSVInf(divided);
        for (int j = 0; j < Columcount; j++)
        {
            CString a = Array->GetAt(j);
            p[i][j] = Array->GetAt(j);
        }
    }

    return p;

}
```