杰迅鸿翔电力高压线接线图软件V1.0

源代码

北京杰迅鸿翔信息技术有限公司

#pragma once

class CBase

{

public:

CBase();

virtual ~CBase();

int dpiX;

int dpiY;

virtual void draw(CDC \* pDC, int zoom) = 0;

virtual void drawByGdiplus(Graphics \*pGraph, int zoom) = 0;

COLORREF m\_color;

int m\_lineWidth;

};

#include "stdafx.h"

#include "GraphBase.h"

CBase::CBase()

{

m\_color = RGB(0, 0, 255);

dpiX = 96;

dpiY = 96;

m\_lineWidth = 1;

}

CBase::~CBase()

{

}

#pragma once

#include "Station.h"

class CLabel

{

public:

CLabel();

~CLabel();

int m\_x;

int m\_y;

int m\_left;

int m\_top;

int m\_right;

int m\_bottom;

CString text;

LOGFONT m\_logFont;

CStation \*m\_pStation;

void draw(CDC \* pDC, float zoom);

void drawByGdiplus(Graphics \*pGraph, int zoom);

bool isInRect(CPoint point);

// 是否被选中

BOOL m\_isSelected;

// 0-场站名称，1-场站电压显示

int m\_labelType;

// 电压显示有名值小数位数

// int m\_nDec;

};

#pragma once

#include "GraphBase.h"

class CCycle :

public CBase

{

public:

CCycle();

~CCycle();

int m\_left;

int m\_top;

int m\_right;

int m\_bottom;

//填充，0-不填充， 1-用指定颜色填充， 2-用白色填充

int m\_fillType;

//填充颜色， m\_fillType == 1有效

COLORREF m\_fillColor;

void draw(CDC \* pDC, int zoom);

void drawByGdiplus(Graphics \*pGraph, int zoom);

};

#include "stdafx.h"

#include "GraphLabel.h"

CLabel::CLabel()

: m\_left(0)

, m\_top(0)

, m\_right(0)

, m\_bottom(0)

, m\_isSelected(FALSE)

, m\_labelType(0)

{

m\_x = 0;

m\_y = 0;

text = L"";

m\_pStation = NULL;

memset(&m\_logFont, 0x0, sizeof(LOGFONT));

}

CLabel::~CLabel()

{

}

void CLabel::draw(CDC \* pDC, float zoom)

{

if (m\_pStation == NULL)

return;

SIZE strSize;

COLORREF textColor = m\_pStation->getGraph()->m\_colorref;

if (m\_pStation->m\_isSelected) {

if (textColor == RGB(0, 0, 0)) {

textColor = RGB(255, 0, 0);

}

else {

textColor = RGB(255, 255, 255) - textColor;

}

}

if (m\_isSelected) {

if (textColor == RGB(0, 0, 0)) {

textColor = RGB(255, 0, 0);

}

else {

textColor = RGB(255, 255, 255) - textColor;

}

}

COLORREF oldTextColor = pDC->SetTextColor(textColor);

// pDC->SetTextColor(RGB(0, 0, 255));

//text; // = m\_pStation->m\_stationName;

int x = m\_pStation->getGraph()->m\_x + m\_x;

int y = m\_pStation->getGraph()->m\_y + m\_y;

LOGFONT logFont;

memcpy(&logFont, &m\_logFont, sizeof(LOGFONT));

logFont.lfHeight = m\_logFont.lfHeight \* zoom / 100.0f;

logFont.lfWidth = m\_logFont.lfWidth \* zoom / 100.0f;

CFont font1;

font1.CreateFontIndirectW(&logFont);

CFont \*pOldFont = pDC->SelectObject(&font1);

GetTextExtentExPoint(pDC->GetSafeHdc(),

text,

text.GetLength(),

100, NULL, NULL, &strSize);

m\_left = x\* zoom / 100.0f;

m\_right = m\_left + strSize.cx;//(x + strSize.cx)\* zoom / 100.0f;

m\_top = (y) \* zoom / 100.0f;

m\_bottom = m\_top + strSize.cy; //(y + strSize.cy) \* zoom / 100.0f;

int oldBkMode = pDC->SetBkMode(TRANSPARENT);

/\*if (m\_pStation->m\_isSelected) {

pDC->SetTextColor(RGB(0, 255, 0));

}\*/

CRect rect;

rect.left = m\_left;

rect.right = m\_right;

rect.top = m\_top;

rect.bottom = m\_top + 2 \* strSize.cy;

//COLORREF color = pDC->SetTextColor(RGB(0, 0, 0));

pDC->DrawText(text, &rect, DT\_CENTER);

//pDC->TextOutW(x \* zoom / 100.0f, (y)\*zoom / 100.0f, text);

pDC->SetTextColor(oldTextColor);

//pDC->SetTextColor(color);

pDC->SetBkMode(oldBkMode);

pDC->SelectObject(pOldFont);

font1.DeleteObject();

}

void CLabel::drawByGdiplus(Graphics \* pGraph, int zoom)

{

if (m\_pStation == NULL)

return;

int x = m\_pStation->getGraph()->m\_x + m\_x;

int y = m\_pStation->getGraph()->m\_y + m\_y;

LOGFONT logFont;

memcpy(&logFont, &m\_logFont, sizeof(LOGFONT));

logFont.lfHeight = m\_logFont.lfHeight \* zoom / 100.0f;

logFont.lfWidth = m\_logFont.lfWidth \* zoom / 100.0f;

Gdiplus::Font font(pGraph->GetHDC(), &logFont);

Color color(0, 0, 255);

if (m\_isSelected) {

color.SetFromCOLORREF(RGB(0, 255, 0));

}

SolidBrush brush(color);

pGraph->DrawString(text, text.GetLength(), &font, PointF(x\* zoom / 100.0f, y \* zoom / 100.0f), &brush);

RectF strSize;

pGraph->MeasureString(text, text.GetLength(), &font, PointF(x\* zoom / 100.0f, y \* zoom / 100.0f), &strSize);

m\_left = strSize.GetLeft();

m\_right = strSize.GetRight();

m\_top = strSize.GetTop();

m\_bottom = strSize.GetBottom();

}

bool CLabel::isInRect(CPoint point)

{

int width = m\_right - m\_left;

int height = m\_bottom - m\_top;

CRect rect;

rect.bottom = m\_bottom;

rect.top = m\_top;

rect.left = m\_left;

rect.right = m\_right;

return PtInRect(&rect, point);

}

#include "stdafx.h"

#include "GraphCycle.h"

CCycle::CCycle()

{

m\_left = 0;

m\_top = 0;

m\_right = 0;

m\_bottom = 0;

m\_fillType = 0;

m\_fillColor = RGB(255, 255, 255);

}

CCycle::~CCycle()

{

}

void CCycle::draw(CDC \* pDC, int zoom)

{

CBrush \* pOldBrush = NULL;

CPen pen(PS\_SOLID, m\_lineWidth, m\_color);

CPen \*pOldPen = pDC->SelectObject(&pen);

CBrush brush(m\_fillColor);

if (m\_fillType == 0) {

pOldBrush = (CBrush \*)pDC->SelectStockObject(NULL\_BRUSH);

}

else {

pOldBrush = pDC->SelectObject(&brush);

}

int dpiX = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSX);

int dpiY = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSY);

int left = m\_left / 72.0f \* dpiX;

int top = m\_top / 72.0f \* dpiY;

int right = m\_right / 72.0f \* dpiX;

int bottom = m\_bottom / 72.0f \* dpiY;

pDC->Ellipse(left\*zoom / 100.0f, top\*zoom / 100.0f, right\*zoom / 100.0f, bottom\*zoom / 100.0f);

pDC->SelectObject(pOldBrush);

pDC->SelectObject(pOldPen);

//测试

/\*CBrush brush1(RGB(0, 0, 0));

pOldBrush = pDC->SelectObject(&brush);

pDC->Rectangle(-5, -5, 5, 5);

pDC->SelectObject(pOldBrush);\*/

}

void CCycle::drawByGdiplus(Graphics \* pGraph, int zoom)

{

REAL xdpi = pGraph->GetDpiX();

REAL ydpi = pGraph->GetDpiY();

int left = m\_left / 72.0f \* xdpi;

int top = m\_top / 72.0f \* ydpi;

int right = m\_right / 72.0f \* xdpi;

int bottom = m\_bottom / 72.0f \* ydpi;

if (m\_fillType == 0) {

Color color;

color.SetFromCOLORREF(m\_color);

Pen pen(color, m\_lineWidth);

pGraph->DrawEllipse(&pen, left\*zoom / 100.0f, top\*zoom / 100.0f, (right - left)\*zoom / 100.0f, (bottom - top)\*zoom / 100.0f);

}

else if (m\_fillType == 2) {

Color color;

color.SetFromCOLORREF(RGB(255, 255, 255));

SolidBrush brush(color);

pGraph->FillEllipse(&brush, left\*zoom / 100.0f, top\*zoom / 100.0f, (right - left)\*zoom / 100.0f, (bottom - top)\*zoom / 100.0f);

}

else {

Color color;

color.SetFromCOLORREF(m\_fillColor);

SolidBrush brush(color);

pGraph->FillEllipse(&brush, left\*zoom / 100.0f, top\*zoom / 100.0f, (right - left)\*zoom / 100.0f, (bottom - top)\*zoom / 100.0f);

}

}

// IpFrame.h : CInPlaceFrame 类的接口

//

#pragma once

class CInPlaceFrame : public COleDocIPFrameWnd

{

DECLARE\_DYNCREATE(CInPlaceFrame)

public:

CInPlaceFrame();

// 特性

public:

// 操作

public:

// 重写

public:

virtual BOOL OnCreateControlBars(CFrameWnd\* pWndFrame, CFrameWnd\* pWndDoc);

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// 实现

public:

virtual ~CInPlaceFrame();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected:

CToolBar m\_wndToolBar;

COleDropTarget m\_dropTarget;

COleResizeBar m\_wndResizeBar;

// 生成的消息映射函数

protected:

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

DECLARE\_MESSAGE\_MAP()

};

// IpFrame.cpp : CInPlaceFrame 类的实现

//

#include "stdafx.h"

#include "EPPEIPSDV3.h"

#include "IpFrame.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CInPlaceFrame

IMPLEMENT\_DYNCREATE(CInPlaceFrame, COleDocIPFrameWnd)

BEGIN\_MESSAGE\_MAP(CInPlaceFrame, COleDocIPFrameWnd)

ON\_WM\_CREATE()

END\_MESSAGE\_MAP()

// CInPlaceFrame 构造/析构

CInPlaceFrame::CInPlaceFrame()

{

}

CInPlaceFrame::~CInPlaceFrame()

{

}

int CInPlaceFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (COleDocIPFrameWnd::OnCreate(lpCreateStruct) == -1)

return -1;

// CResizeBar 实现就地调整大小。

if (!m\_wndResizeBar.Create(this))

{

TRACE0("Failed to create resize bar\n");

return -1; // 未能创建

}

// 默认情况下，最好注册一个与框架窗口

// 无关的放置目标。 这将防止

// 拖放对象“错过”支持拖放的容器中。

m\_dropTarget.Register(this);

return 0;

}

// 框架将调用 OnCreateControlBars 以便在

// 容器应用程序的窗口上创建控件条。 pWndFrame 是容器的顶级框架窗口，

// 并且始终不为 NULL。 pWndDoc 是文档级框架窗口，

// 并且在容器是 SDI 应用程序时为 NULL。

// 服务器应用程序可以将 MFC 控件条放在这两个窗口中的任何一个上。

BOOL CInPlaceFrame::OnCreateControlBars(CFrameWnd\* pWndFrame, CFrameWnd\* pWndDoc)

{

// 如果使用 pWndDoc，则将此移除

UNREFERENCED\_PARAMETER(pWndDoc);

// 设置此窗口的所有者，以便将消息发送到正确的应用程序

m\_wndToolBar.SetOwner(this);

// 在客户端的框架窗口上创建工具栏

if (!m\_wndToolBar.CreateEx(pWndFrame, TBSTYLE\_FLAT,WS\_CHILD | WS\_VISIBLE | CBRS\_TOP

| CBRS\_GRIPPER | CBRS\_TOOLTIPS | CBRS\_FLYBY | CBRS\_SIZE\_DYNAMIC) ||

!m\_wndToolBar.LoadToolBar(IDR\_EPPEIPSDTYPE\_SRVR\_IP))

{

TRACE0("Failed to create toolbar\n");

return FALSE;

}

// TODO: 如果不需要可停靠工具栏，则删除这三行

m\_wndToolBar.EnableDocking(CBRS\_ALIGN\_ANY);

pWndFrame->EnableDocking(CBRS\_ALIGN\_ANY);

pWndFrame->DockControlBar(&m\_wndToolBar);

return TRUE;

}

BOOL CInPlaceFrame::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: 在此处通过修改 CREATESTRUCT cs 来修改窗口类或样式

return COleDocIPFrameWnd::PreCreateWindow(cs);

}

// CInPlaceFrame 诊断

#ifdef \_DEBUG

void CInPlaceFrame::AssertValid() const

{

COleDocIPFrameWnd::AssertValid();

}

void CInPlaceFrame::Dump(CDumpContext& dc) const

{

COleDocIPFrameWnd::Dump(dc);

}

#endif //\_DEBUG

// CInPlaceFrame 命令

#pragma once

class CInterareaTransferData

{

public:

CInterareaTransferData();

~CInterareaTransferData();

int ARFROM;

int ARTO;

CString TRID;

float PTRAN;

};

#include "stdafx.h"

#include "InterareaTransferData.h"

CInterareaTransferData::CInterareaTransferData()

{

ARFROM = 0;

ARTO = 0;

TRID = \_T("");

PTRAN = 0.0f;

}

CInterareaTransferData::~CInterareaTransferData()

{

}

#pragma once

#include "GraphBase.h"

class CRect : public CBase

{

public:

CRect();

~CRect();

void draw(CDC \* pDC, int zoom);

int m\_left;

int m\_top;

int m\_right;

int m\_bottom;

//填充，0-不填充， 1-用指定颜色填充， 2-用白色填充

int m\_fillType;

//填充颜色， m\_fillType == 1有效

COLORREF m\_fillColor;

void drawByGdiplus(Graphics \*pGraph, int zoom);

};

#include "stdafx.h"

#include "GraphRect.h"

CRect::CRect()

{

m\_left = 0;

m\_top = 0;

m\_right = 0;

m\_bottom = 0;

m\_fillType = 0;

m\_fillColor = RGB(255, 255, 255);

}

CRect::~CRect()

{

}

void CRect::draw(CDC \* pDC, int zoom)

{

CBrush \* pOldBrush = NULL;

CPen pen(PS\_SOLID, m\_lineWidth, m\_color);

CPen \*pOldPen = pDC->SelectObject(&pen);

CBrush brush(m\_fillColor);

if (m\_fillType == 0) {

pOldBrush = (CBrush \*)pDC->SelectStockObject(NULL\_BRUSH);

}

else{

pOldBrush = pDC->SelectObject(&brush);

}

//int dpiX = pDC->GetDeviceCaps(LOGPIXELSX);

int dpiX = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSX);

int dpiY = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSY);

// int dpiY = pDC->GetDeviceCaps(LOGPIXELSY);

int left = m\_left / 72.0f \* dpiX;

int top = m\_top / 72.0f \* dpiY;

int right = m\_right / 72.0f \* dpiX;

int bottom = m\_bottom / 72.0f \* dpiY;

pDC->Rectangle(left\*zoom / 100.0f, top\*zoom / 100.0f, right\*zoom / 100.0f, bottom\*zoom / 100.0f);

pDC->SelectObject(pOldBrush);

pDC->SelectObject(pOldPen);

}

void CRect::drawByGdiplus(Graphics \* pGraph, int zoom)

{

Color color;

color.SetFromCOLORREF(m\_color);

Pen pen(color, m\_lineWidth);

int dpiX = pGraph->GetDpiX();

int dpiY = pGraph->GetDpiY();

int left = m\_left / 72.0f \* dpiX;

int top = m\_top / 72.0f \* dpiY;

int right = m\_right / 72.0f \* dpiX;

int bottom = m\_bottom / 72.0f \* dpiY;

pGraph->DrawRectangle(&pen, Rect(left \* zoom / 100.0f, top \* zoom / 100.0f, (right - left)\*zoom / 100.0f, (bottom - top)\* zoom / 100.0f));

if (m\_fillType != 0) {

Color color;

if (m\_fillType == 1) {

color.SetFromCOLORREF(m\_fillColor);

}

else {

color.SetFromCOLORREF(RGB(255, 255, 255));

}

SolidBrush brush(color);

pGraph->FillRectangle(&brush, Rect(left \* zoom / 100.0f, top \* zoom / 100.0f, (right - left)\*zoom / 100.0f, (bottom - top)\* zoom / 100.0f));

}

}

#pragma once

#include "GraphBase.h"

#include <vector>

using namespace std;

class CPolygon :

public CBase

{

public:

CPolygon();

~CPolygon();

void draw(CDC \* pDC, int zoom);

vector<CPoint> m\_points;

//填充，0-不填充， 1-用指定颜色填充， 2-用白色填充

int m\_fillType;

//填充颜色， m\_fillType == 1有效

COLORREF m\_fillColor;

void drawByGdiplus(Graphics \*pGraph, int zoom);

};

#include "stdafx.h"

#include "GraphPolygon.h"

CPolygon::CPolygon()

: m\_fillType(0)

{

m\_fillColor = RGB(255, 255, 255);

}

CPolygon::~CPolygon()

{

}

void CPolygon::draw(CDC \* pDC, int zoom)

{

int size = m\_points.size();

if (size < 3)

return;

CBrush \* pOldBrush = NULL;

CPen pen(PS\_SOLID, m\_lineWidth, m\_color);

CPen \*pOldPen = pDC->SelectObject(&pen);

CBrush brush(m\_fillColor);

if (m\_fillType == 0) {

pOldBrush = (CBrush \*)pDC->SelectStockObject(NULL\_BRUSH);

}

else {

pOldBrush = pDC->SelectObject(&brush);

}

int dpiX = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSX);

int dpiY = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSY);

POINT \*point = new POINT[size];

for (int i = 0; i < size;i++) {

CPoint p = m\_points[i];

point[i].x = p.x / 72.0f \* dpiX \* zoom / 100.0f;

point[i].y = p.y / 72.0f \* dpiY \* zoom / 100.0f;

}

pDC->Polygon(point, size);

delete[] point;

pDC->SelectObject(pOldBrush);

pDC->SelectObject(pOldPen);

}

void CPolygon::drawByGdiplus(Graphics \* pGraph, int zoom)

{

int size = m\_points.size();

if (size < 3)

return;

Color color;

color.SetFromCOLORREF(m\_color);

Pen pen(color, m\_lineWidth);

int dpiX = pGraph->GetDpiX();

int dpiY = pGraph->GetDpiY();

Point \*pPoints = new Point[size];

for (int i = 0; i < size; i++) {

pPoints[i].X = m\_points[i].x / 72.0f \* dpiX \* zoom / 100.0f;

pPoints[i].Y = m\_points[i].y / 72.0f \* dpiY \* zoom / 100.0f;

}

pGraph->DrawPolygon(&pen, pPoints, size);

if (m\_fillType != 0) {

Color color;

if (m\_fillType == 1) {

color.SetFromCOLORREF(m\_fillColor);

}

else {

color.SetFromCOLORREF(RGB(255, 255, 255));

}

SolidBrush brush(color);

pGraph->FillPolygon(&brush, pPoints, size);

}

delete[] pPoints;

pPoints = nullptr;

}

#pragma once

#include "GraphBase.h"

class CLine : public CBase

{

public:

CLine();

~CLine();

void draw(CDC \* pDC, int zoom);

// 0-实线，1-虚线， 2-点线， 3-点划线，4-双点划线

int m\_type;

int m\_startX;

int m\_startY;

int m\_endX;

int m\_endY;

//线宽， 0为默认线宽（1），如果宽度大于2，线性被忽略。

int m\_lineWidth1;

void drawByGdiplus(Graphics \*pGraph, int zoom);

};

#include "stdafx.h"

#include "GraphLine.h"

CLine::CLine()

: m\_type(0)

, m\_startX(0)

{

m\_startY = 0;

m\_endX = 0;

m\_endY = 0;

m\_lineWidth1 = 0;

}

CLine::~CLine()

{

}

void CLine::draw(CDC \* pDC, int zoom) {

ASSERT\_VALID(pDC);

CPen \*pOldPen = NULL;

int lineType = PS\_SOLID;

if (m\_lineWidth1 > 1) {

lineType = PS\_SOLID;

}

else {

switch (m\_type)

{

case 0:

lineType = PS\_SOLID;

break;

case 1:

lineType = PS\_DASH;

break;

case 2:

lineType = PS\_DOT;

break;

case 3:

lineType = PS\_DASHDOT;

break;

case 4:

lineType = PS\_DASHDOTDOT;

break;

default:

lineType = PS\_SOLID;

break;

}

}

int dpiX = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSX);

int dpiY = GetDeviceCaps(pDC->m\_hDC, LOGPIXELSY);

int x0 = m\_startX / 72.0f \* dpiX;

int y0 = m\_startY / 72.0f \* dpiY;

int x1 = m\_endX / 72.0f \* dpiX;

int y1 = m\_endY / 72.0f \* dpiY;

CPen pen(lineType, m\_lineWidth1, m\_color);

pOldPen = pDC->SelectObject(&pen);

pDC->MoveTo(x0 \* zoom/100.0f, y0\*zoom/100.0f);

pDC->LineTo(x1\*zoom / 100.0f, y1\*zoom / 100.0f);

pDC->SelectObject(pOldPen);

}

void CLine::drawByGdiplus(Graphics \* pGraph, int zoom)

{

Color color;

color.SetFromCOLORREF(m\_color);

Pen pen(color, m\_lineWidth1);

switch (m\_type)

{

case 0:

pen.SetDashStyle(DashStyleSolid);

break;

case 1:

pen.SetDashStyle(DashStyleDash);

break;

case 2:

pen.SetDashStyle(DashStyleDashDot);

break;

case 3:

pen.SetDashStyle(DashStyleDashDotDot);

break;

default:

pen.SetDashStyle(DashStyleSolid);

break;

}

int dpiX = pGraph->GetDpiX();

int dpiY = pGraph->GetDpiY();

int x0 = m\_startX / 72.0f \* dpiX;

int y0 = m\_startY / 72.0f \* dpiY;

int x1 = m\_endX / 72.0f \* dpiX;

int y1 = m\_endY / 72.0f \* dpiY;

pGraph->DrawLine(&pen, Point(x0 \* zoom / 100.0f, y0 \* zoom / 100.0f), Point(x1 \* zoom / 100.0f, y1 \* zoom / 100.0f));

}

#pragma once

#include "afxwin.h"

// CControlPanel 对话框

class CControlPanel : public CDialogEx

{

DECLARE\_DYNAMIC(CControlPanel)

public:

CControlPanel(CWnd\* pParent = NULL); // 标准构造函数

virtual ~CControlPanel();

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_DIALOG\_CONTROL\_PANEL };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnViewControlPanel();

int m\_graphType;

BOOL m\_isDisplayStationName;

BOOL m\_isDisplayLine;

BOOL m\_isDisplayMWAR;

BOOL m\_isAngle;

// CButton m\_isDiaplayStationData;

BOOL m\_isDispalyLineData;

// BOOL m\_isDisplaySelf;

int m\_nDisplayUnit;

BOOL m\_isDisplayStationData;

BOOL m\_isDisplayShunt;

BOOL m\_isDisplaySelf;

};

// ControlPanel.cpp : 实现文件

//

#include "stdafx.h"

#include "EPPEIPSDV3.h"

#include "ControlPanel.h"

#include "afxdialogex.h"

// CControlPanel 对话框

IMPLEMENT\_DYNAMIC(CControlPanel, CDialogEx)

CControlPanel::CControlPanel(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(IDD\_DIALOG\_CONTROL\_PANEL, pParent)

, m\_graphType(0)

, m\_isDisplayStationName(FALSE)

, m\_isDisplayLine(FALSE)

, m\_isDisplayMWAR(FALSE)

, m\_isAngle(FALSE)

, m\_isDispalyLineData(FALSE)

, m\_isDisplaySelf(FALSE)

, m\_nDisplayUnit(0)

, m\_isDisplayStationData(FALSE)

, m\_isDisplayShunt(FALSE)

{

}

CControlPanel::~CControlPanel()

{

}

void CControlPanel::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Radio(pDX, IDC\_RADIO1, m\_graphType);

DDX\_Check(pDX, IDC\_CHECK1, m\_isDisplayStationName);

DDX\_Check(pDX, IDC\_CHECK7, m\_isDisplayLine);

DDX\_Check(pDX, IDC\_CHECK9, m\_isDisplayMWAR);

DDX\_Check(pDX, IDC\_CHECK11, m\_isAngle);

// DDX\_Control(pDX, IDC\_CHECK6, m\_isDiaplayStationData);

DDX\_Check(pDX, IDC\_CHECK8, m\_isDispalyLineData);

// DDX\_Check(pDX, IDC\_CHECK10, m\_isDisplaySelf);

DDX\_Radio(pDX, IDC\_RADIO2, m\_nDisplayUnit);

DDX\_Check(pDX, IDC\_CHECK6, m\_isDisplayStationData);

DDX\_Check(pDX, IDC\_CHECK10, m\_isDisplayShunt);

DDX\_Check(pDX, IDC\_CHECK12, m\_isDisplaySelf);

}

BEGIN\_MESSAGE\_MAP(CControlPanel, CDialogEx)

ON\_COMMAND(ID\_VIEW\_CONTROL\_PANEL, &CControlPanel::OnViewControlPanel)

END\_MESSAGE\_MAP()

// CControlPanel 消息处理程序

//控制板

void CControlPanel::OnViewControlPanel()

{

// TODO: 在此添加命令处理程序代码

}

#pragma once

// CColorListCtrl

class CColorListCtrl : public CListCtrl

{

DECLARE\_DYNAMIC(CColorListCtrl)

public:

CColorListCtrl();

virtual ~CColorListCtrl();

CMap<int, int, COLORREF, COLORREF> colorMap;

protected:

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnNMDblclk(NMHDR \*pNMHDR, LRESULT \*pResult);

afx\_msg void OnNMCustomdraw(NMHDR \*pNMHDR, LRESULT \*pResult);

};

// ColorListCtrl.cpp : 实现文件

//

#include "stdafx.h"

#include "EPPEIPSDV3.h"

#include "ColorListCtrl.h"

// CColorListCtrl

IMPLEMENT\_DYNAMIC(CColorListCtrl, CListCtrl)

CColorListCtrl::CColorListCtrl()

{

}

CColorListCtrl::~CColorListCtrl()

{

}

BEGIN\_MESSAGE\_MAP(CColorListCtrl, CListCtrl)

ON\_NOTIFY\_REFLECT(NM\_DBLCLK, &CColorListCtrl::OnNMDblclk)

ON\_NOTIFY\_REFLECT(NM\_CUSTOMDRAW, &CColorListCtrl::OnNMCustomdraw)

END\_MESSAGE\_MAP()

// CColorListCtrl 消息处理程序

void CColorListCtrl::OnNMDblclk(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMITEMACTIVATE pNMItemActivate = reinterpret\_cast<LPNMITEMACTIVATE>(pNMHDR);

// TODO: 在此添加控件通知处理程序代码

\*pResult = 0;

}

void CColorListCtrl::OnNMCustomdraw(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMCUSTOMDRAW pNMCD = reinterpret\_cast<LPNMCUSTOMDRAW>(pNMHDR);

// TODO: 在此添加控件通知处理程序代码

\*pResult = CDRF\_DODEFAULT;;

NMLVCUSTOMDRAW \* lplvdr = (NMLVCUSTOMDRAW\*)pNMHDR;

NMCUSTOMDRAW &nmcd = lplvdr->nmcd;

switch (lplvdr->nmcd.dwDrawStage)//判断状态

{

case CDDS\_PREPAINT:

{

\*pResult = CDRF\_NOTIFYITEMDRAW;

break;

}

case CDDS\_ITEMPREPAINT:

{

\*pResult = CDRF\_NOTIFYSUBITEMDRAW;

break;

}

case (CDDS\_ITEMPREPAINT | CDDS\_SUBITEM)://如果为画ITEM之前就要进行颜色的改变

{

COLORREF ItemColor;

if (colorMap.Lookup(nmcd.dwItemSpec, ItemColor))

//根据在SetItemColor(DWORD iItem, COLORREF color) 设置的//ITEM号和COLORREF 在摸板中查找，然后进行颜色赋值。

{

if (lplvdr->iSubItem == 1) {

lplvdr->clrTextBk = ItemColor;

}

\*pResult = CDRF\_DODEFAULT;

}

}

break;

}

}

// ChildFrm.h : CChildFrame 类的接口

//

#pragma once

class CChildFrame : public CMDIChildWndEx

{

DECLARE\_DYNCREATE(CChildFrame)

public:

CChildFrame();

// 特性

public:

// 操作

public:

// 重写

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// 实现

public:

virtual ~CChildFrame();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

// 生成的消息映射函数

protected:

DECLARE\_MESSAGE\_MAP()

};

// ChildFrm.cpp : CChildFrame 类的实现

//

#include "stdafx.h"

#include "EPPEIPSDV3.h"

#include "ChildFrm.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CChildFrame

IMPLEMENT\_DYNCREATE(CChildFrame, CMDIChildWndEx)

BEGIN\_MESSAGE\_MAP(CChildFrame, CMDIChildWndEx)

END\_MESSAGE\_MAP()

// CChildFrame 构造/析构

CChildFrame::CChildFrame()

{

// TODO: 在此添加成员初始化代码

}

CChildFrame::~CChildFrame()

{

}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: 在此处通过修改 CREATESTRUCT cs 来修改窗口类或样式

if( !CMDIChildWndEx::PreCreateWindow(cs) )

return FALSE;

return TRUE;

}

// CChildFrame 诊断

#ifdef \_DEBUG

void CChildFrame::AssertValid() const

{

CMDIChildWndEx::AssertValid();

}

void CChildFrame::Dump(CDumpContext& dc) const

{

CMDIChildWndEx::Dump(dc);

}

#endif //\_DEBUG

// CChildFrame 消息处理程序

#pragma once

class CControlPoint

{

public:

CControlPoint();

~CControlPoint();

};

#include "stdafx.h"

#include "ControlPoint.h"

CControlPoint::CControlPoint()

{

}

CControlPoint::~CControlPoint()

{

}

#include "stdafx.h"

#include "MemoText.h"

CMemoText::CMemoText()

: m\_nAlgin(0)

, m\_isShowLine(false)

, m\_text(\_T(""))

, m\_x(0)

, m\_y(0)

, m\_nHeight(0)

, m\_nWidth(0)

, m\_isSelected(false)

{

memset(&m\_logfont, 0x0, sizeof(LOGFONT));

m\_rectTracker.m\_nStyle = CRectTracker::resizeInside | CRectTracker::dottedLine;//设置RectTracker样式

m\_rectTracker.m\_nHandleSize = 5; //控制柄的像素大小

m\_rectTracker.m\_rect.SetRect(-1, -1, -1, -1); //初始化m\_rect的值

}

CMemoText::CMemoText(POINT point, CString text, bool isShowLine, int align, COLORREF colorref, LOGFONT & logfont)

{

m\_isSelected = false;

m\_nHeight = 0;

m\_nWidth = 0;

m\_x = point.x;

m\_y = point.y;

m\_text = text;

m\_isShowLine = isShowLine;

m\_nAlgin = align;

m\_colorref = colorref;

memcpy(&m\_logfont, &logfont, sizeof(LOGFONT));

}

CMemoText::~CMemoText()

{

}

void CMemoText::draw(CDC \* pDC, float zoom)

{

SIZE strSize1;

CFont font;

font.CreateFontIndirectW(&m\_logfont);

CFont \*pOld = pDC->SelectObject(&font);

GetTextExtentExPoint(pDC->GetSafeHdc(),

m\_text,

m\_text.GetLength(),

100, NULL, NULL, &strSize1);

//m\_nWidth = strSize1.cx + 3;

//m\_nHeight = strSize1.cy + 3;

pDC->SelectObject(pOld);

font.DeleteObject();

int count = 1;

int pos = m\_text.Find(L"\n");

while (pos != -1) {

count++;

pos = m\_text.Find(L"\n", pos + 1);

}

if ((m\_nWidth < strSize1.cx ) || (m\_nHeight < (count \* strSize1.cy))) {

m\_nWidth = strSize1.cx + 3;

m\_nHeight = count \* strSize1.cy + 3;

}

SIZE strSize;

pDC->SetTextColor(m\_colorref);

//text; // = m\_pStation->m\_stationName;

int x = m\_x;

int y = m\_y;

LOGFONT logFont;

memcpy(&logFont, &m\_logfont, sizeof(LOGFONT));

logFont.lfHeight = m\_logfont.lfHeight \* zoom / 100.0f;

logFont.lfWidth = m\_logfont.lfWidth \* zoom / 100.0f;

CFont font1;

font1.CreateFontIndirectW(&logFont);

CFont \*pOldFont = pDC->SelectObject(&font1);

GetTextExtentExPoint(pDC->GetSafeHdc(),

m\_text,

m\_text.GetLength(),

100, NULL, NULL, &strSize);

int oldBkMode = pDC->SetBkMode(TRANSPARENT);

if (m\_isSelected) {

pDC->SetTextColor(RGB(0, 255, 0));

}

//CRect realRect;

realRect.left = x \* zoom / 100.f;

realRect.right = (x + m\_nWidth) \* zoom / 100.0f;

realRect.top = y \* zoom / 100.0f;

realRect.bottom = (y + m\_nHeight) \* zoom / 100.0f;

//COLORREF color = pDC->SetTextColor(RGB(0, 0, 0));

if (m\_isShowLine) {

CBrush \* pOldBrush = (CBrush \*)pDC->SelectStockObject(NULL\_BRUSH);

pDC->Rectangle(&realRect);

pDC->SelectObject(pOldBrush);

}

UINT format = DT\_CENTER;

switch (m\_nAlgin)

{

case 0:

format = DT\_LEFT;

break;

case 1:

format = DT\_CENTER;

break;

case 2:

format = DT\_RIGHT;

break;

default:

format = DT\_LEFT;

break;

}

pDC->DrawText(m\_text, &realRect, format);

//pDC->TextOutW(x \* zoom / 100.0f, (y)\*zoom / 100.0f, text);

//pDC->SetTextColor(color);

pDC->SetBkMode(oldBkMode);

pDC->SelectObject(pOldFont);

font1.DeleteObject();

}

bool CMemoText::isPtIn(CPoint & point)

{

if (realRect.PtInRect(point)) {

m\_isSelected = true;

return true;

}

else {

m\_isSelected = false;

return false;

}

// return realRect.PtInRect(point);

// return false;

}

// MainFrm.h : CMainFrame 类的接口

//

#pragma once

class CMainFrame : public CMDIFrameWndEx

{

DECLARE\_DYNAMIC(CMainFrame)

public:

CMainFrame();

// 特性

public:

// 操作

public:

int UpdateZoom(CString & text);

// 重写

public:

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// 实现

public:

virtual ~CMainFrame();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected: // 控件条嵌入成员

CMFCMenuBar m\_wndMenuBar;

CMFCToolBar m\_wndToolBar;

CMFCStatusBar m\_wndStatusBar;

// 生成的消息映射函数

protected:

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

afx\_msg void OnWindowManager();

afx\_msg void OnViewCustomize();

afx\_msg LRESULT OnToolbarCreateNew(WPARAM wp, LPARAM lp);

afx\_msg void OnApplicationLook(UINT id);

afx\_msg void OnUpdateApplicationLook(CCmdUI\* pCmdUI);

DECLARE\_MESSAGE\_MAP()

};

// MainFrm.cpp : CMainFrame 类的实现

//

#include "stdafx.h"

#include "EPPEIPSDV3.h"

#include "MainFrm.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CMainFrame

IMPLEMENT\_DYNAMIC(CMainFrame, CMDIFrameWndEx)

BEGIN\_MESSAGE\_MAP(CMainFrame, CMDIFrameWndEx)

ON\_WM\_CREATE()

ON\_COMMAND(ID\_WINDOW\_MANAGER, &CMainFrame::OnWindowManager)

ON\_COMMAND(ID\_VIEW\_CUSTOMIZE, &CMainFrame::OnViewCustomize)

ON\_REGISTERED\_MESSAGE(AFX\_WM\_CREATETOOLBAR, &CMainFrame::OnToolbarCreateNew)

ON\_COMMAND\_RANGE(ID\_VIEW\_APPLOOK\_WIN\_2000, ID\_VIEW\_APPLOOK\_WINDOWS\_7, &CMainFrame::OnApplicationLook)

ON\_UPDATE\_COMMAND\_UI\_RANGE(ID\_VIEW\_APPLOOK\_WIN\_2000, ID\_VIEW\_APPLOOK\_WINDOWS\_7, &CMainFrame::OnUpdateApplicationLook)

END\_MESSAGE\_MAP()

static UINT indicators[] =

{

ID\_SEPARATOR, // 状态行指示器

ID\_INDICATOR\_CAPS,

ID\_INDICATOR\_NUM,

ID\_INDICATOR\_SCRL,

};

// CMainFrame 构造/析构

CMainFrame::CMainFrame()

{

// TODO: 在此添加成员初始化代码

theApp.m\_nAppLook = theApp.GetInt(\_T("ApplicationLook"), ID\_VIEW\_APPLOOK\_VS\_2008);

}

CMainFrame::~CMainFrame()

{

}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CMDIFrameWndEx::OnCreate(lpCreateStruct) == -1)

return -1;

BOOL bNameValid;

if (!m\_wndMenuBar.Create(this))

{

TRACE0("未能创建菜单栏\n");

return -1; // 未能创建

}

m\_wndMenuBar.SetPaneStyle(m\_wndMenuBar.GetPaneStyle() | CBRS\_SIZE\_DYNAMIC | CBRS\_TOOLTIPS | CBRS\_FLYBY);

// 防止菜单栏在激活时获得焦点

CMFCPopupMenu::SetForceMenuFocus(FALSE);

if (!m\_wndToolBar.CreateEx(this, TBSTYLE\_FLAT, WS\_CHILD | WS\_VISIBLE | CBRS\_TOP | CBRS\_GRIPPER | CBRS\_TOOLTIPS | CBRS\_FLYBY | CBRS\_SIZE\_DYNAMIC) ||

!m\_wndToolBar.LoadToolBar(theApp.m\_bHiColorIcons ? IDR\_MAINFRAME\_24 : IDR\_MAINFRAME))

{

TRACE0("未能创建工具栏\n");

return -1; // 未能创建

}

CString strToolBarName;

bNameValid = strToolBarName.LoadString(IDS\_TOOLBAR\_STANDARD);

ASSERT(bNameValid);

m\_wndToolBar.SetWindowText(strToolBarName);

CString strCustomize;

bNameValid = strCustomize.LoadString(IDS\_TOOLBAR\_CUSTOMIZE);

ASSERT(bNameValid);

m\_wndToolBar.EnableCustomizeButton(TRUE, ID\_VIEW\_CUSTOMIZE, strCustomize);

if (!m\_wndStatusBar.Create(this))

{

TRACE0("未能创建状态栏\n");

return -1; // 未能创建

}

m\_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));

// TODO: 如果您不希望工具栏和菜单栏可停靠，请删除这五行

m\_wndMenuBar.EnableDocking(CBRS\_ALIGN\_ANY);

m\_wndToolBar.EnableDocking(CBRS\_ALIGN\_ANY);

EnableDocking(CBRS\_ALIGN\_ANY);

DockPane(&m\_wndMenuBar);

DockPane(&m\_wndToolBar);

// 启用 Visual Studio 2005 样式停靠窗口行为

CDockingManager::SetDockingMode(DT\_SMART);

// 启用 Visual Studio 2005 样式停靠窗口自动隐藏行为

EnableAutoHidePanes(CBRS\_ALIGN\_ANY);

// 基于持久值设置视觉管理器和样式

OnApplicationLook(theApp.m\_nAppLook);

// 启用增强的窗口管理对话框

EnableWindowsDialog(ID\_WINDOW\_MANAGER, ID\_WINDOW\_MANAGER, TRUE);

// 启用工具栏和停靠窗口菜单替换

EnablePaneMenu(TRUE, ID\_VIEW\_CUSTOMIZE, strCustomize, ID\_VIEW\_TOOLBAR);

// 启用快速(按住 Alt 拖动)工具栏自定义

CMFCToolBar::EnableQuickCustomization();

return 0;

}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)

{

if( !CMDIFrameWndEx::PreCreateWindow(cs) )

return FALSE;

// TODO: 在此处通过修改

// CREATESTRUCT cs 来修改窗口类或样式

return TRUE;

}

// CMainFrame 诊断

#ifdef \_DEBUG

void CMainFrame::AssertValid() const

{

CMDIFrameWndEx::AssertValid();

}

void CMainFrame::Dump(CDumpContext& dc) const

{

CMDIFrameWndEx::Dump(dc);

}

#endif //\_DEBUG

// CMainFrame 消息处理程序

void CMainFrame::OnWindowManager()

{

ShowWindowsDialog();

}

void CMainFrame::OnViewCustomize()

{

CMFCToolBarsCustomizeDialog\* pDlgCust = new CMFCToolBarsCustomizeDialog(this, TRUE /\* 扫描菜单\*/);

pDlgCust->Create();

}

LRESULT CMainFrame::OnToolbarCreateNew(WPARAM wp,LPARAM lp)

{

LRESULT lres = CMDIFrameWndEx::OnToolbarCreateNew(wp,lp);

if (lres == 0)

{

return 0;

}

CMFCToolBar\* pUserToolbar = (CMFCToolBar\*)lres;

ASSERT\_VALID(pUserToolbar);

BOOL bNameValid;

CString strCustomize;

bNameValid = strCustomize.LoadString(IDS\_TOOLBAR\_CUSTOMIZE);

ASSERT(bNameValid);

pUserToolbar->EnableCustomizeButton(TRUE, ID\_VIEW\_CUSTOMIZE, strCustomize);

return lres;

}

void CMainFrame::OnApplicationLook(UINT id)

{

CWaitCursor wait;

theApp.m\_nAppLook = id;

switch (theApp.m\_nAppLook)

{

case ID\_VIEW\_APPLOOK\_WIN\_2000:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManager));

break;

case ID\_VIEW\_APPLOOK\_OFF\_XP:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOfficeXP));

break;

case ID\_VIEW\_APPLOOK\_WIN\_XP:

CMFCVisualManagerWindows::m\_b3DTabsXPTheme = TRUE;

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));

break;

case ID\_VIEW\_APPLOOK\_OFF\_2003:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOffice2003));

CDockingManager::SetDockingMode(DT\_SMART);

break;

case ID\_VIEW\_APPLOOK\_VS\_2005:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerVS2005));

CDockingManager::SetDockingMode(DT\_SMART);

break;

case ID\_VIEW\_APPLOOK\_VS\_2008:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerVS2008));

CDockingManager::SetDockingMode(DT\_SMART);

break;

case ID\_VIEW\_APPLOOK\_WINDOWS\_7:

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows7));

CDockingManager::SetDockingMode(DT\_SMART);

break;

default:

switch (theApp.m\_nAppLook)

{

case ID\_VIEW\_APPLOOK\_OFF\_2007\_BLUE:

CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_LunaBlue);

break;

case ID\_VIEW\_APPLOOK\_OFF\_2007\_BLACK:

CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_ObsidianBlack);

break;

case ID\_VIEW\_APPLOOK\_OFF\_2007\_SILVER:

CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_Silver);

break;

case ID\_VIEW\_APPLOOK\_OFF\_2007\_AQUA:

CMFCVisualManagerOffice2007::SetStyle(CMFCVisualManagerOffice2007::Office2007\_Aqua);

break;

}

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerOffice2007));

CDockingManager::SetDockingMode(DT\_SMART);

}

RedrawWindow(NULL, NULL, RDW\_ALLCHILDREN | RDW\_INVALIDATE | RDW\_UPDATENOW | RDW\_FRAME | RDW\_ERASE);

theApp.WriteInt(\_T("ApplicationLook"), theApp.m\_nAppLook);

}

void CMainFrame::OnUpdateApplicationLook(CCmdUI\* pCmdUI)

{

pCmdUI->SetRadio(theApp.m\_nAppLook == pCmdUI->m\_nID);

}

int CMainFrame::UpdateZoom(CString & text)

{

m\_wndStatusBar.SetPaneText(1, text);

return 0;

}

#pragma once

class CMemoText

{

public:

CMemoText();

CMemoText(POINT point, CString text, bool isShowLine, int align, COLORREF colorref, LOGFONT &logfont);

~CMemoText();

LOGFONT m\_logfont;

COLORREF m\_colorref;

// 0-左， 1-中， 2-右

int m\_nAlgin;

bool m\_isShowLine;

CString m\_text;

public:

int m\_x;

int m\_y;

void draw(CDC \* pDC, float zoom);

int m\_nHeight;

int m\_nWidth;

bool m\_isSelected;

bool isPtIn(CPoint & point);

CRect realRect;

CRectTracker m\_rectTracker;

};

#pragma once

#include "PSSEBus.h"

class CPSSEBus;

//#include "Edge.h"

class CEdge;

class CEdgePath;

#include <vector>

using namespace std;

//支路

class CPSSEBranch

{

public:

CPSSEBranch();

~CPSSEBranch();

//连接的母线编号，如果两个，就是【0】 -》 【1】

vector<CPSSEBus \*> m\_busPoints;

// 支路类型，0-非变压器支路，1-开关，2-2绕组变压器，3-3绕组变压器,4-双端直流，5-多端直流,6-VSC DC Line

int m\_type;

/\*\*

non transform branche

One- or two-character uppercase non-blank alphanumeric branch circuit identifier;

the first character of CKT must not be an ampersand ( & ); refer to Multi-Section

Line Grouping Data. If the first character of CKT is an at sign ( @ ), the branch is

treated as a breaker; if it is an asterisk ( \* ), it is treated as a switch (see Section

6.15.2, Outage Statistics Data File Contents). If the first character of CKT is greater

than sign (>), the branch buses I and J belong to the same substation in GIC data

(see Section 7.2, GIC Data File Contents). Unless it is a breaker, switch, or branch

in GIC data substation, it is recommended that single circuit branches be designated as having the circuit identifier 1. CKT = 1 by default.

System switch Device

two-character uppercase non-blank alphanumeric switching device

identifier; CKT = 1 by default.

Transformer

One- or two-character uppercase non-blank alphanumeric transformer circuit identifier; the first character of CKT must not be an ampersand ( & ), at sign ( @ ), or

asterisk (  ); refer to Multi-Section Line Grouping Data and Section 6.15.2, Outage

Statistics Data File Contents. CKT = 1 by default.

\*/

CString m\_strCKT;

CString name;

/\*float tobus\_mw;

float tobus\_mwar;

CString tobus\_ratio;\*/

// 所属图形线

CEdge \*m\_pEdge;

//属于哪个分组；

CEdgePath \*m\_pEdgePath;

// 获取唯一标识

virtual CString getIdenetifyName() = 0;

float m\_IBus\_MW;

float m\_IBS\_MWAR;

float m\_JBUS\_MW;

float m\_JBUS\_MWAR;

// 是否已经加载潮流运算结果

BOOL m\_isPowflow;

};

#include "stdafx.h"

#include "PSSEBranch.h"

CPSSEBranch::CPSSEBranch()

: m\_type(0)

, m\_strCKT(\_T(""))

, name(\_T(""))

, m\_pEdge(NULL)

, m\_IBus\_MW(0)

, m\_isPowflow(FALSE)

, m\_pEdgePath(NULL)

{

/\*tobus\_mw = 0.0f;

tobus\_mwar = 0.0f;

tobus\_ratio = L"";\*/

m\_IBS\_MWAR = 0.0f;

m\_JBUS\_MW = 0.0f;

m\_JBUS\_MWAR = 0.0f;

}

CPSSEBranch::~CPSSEBranch()

{

m\_busPoints.clear();

}

#pragma once

#include "GraphBase.h"

#include "Station.h"

#include <vector>

using namespace std;

class CStation;

class CPSDStationGraph

{

public:

CPSDStationGraph();

virtual ~CPSDStationGraph();

// 画图函数 ,x y 为左上角坐标

void draw(CDC \* pDC, int zoom, int x, int y);

void drawSelectedRect(CDC \* pDC, int zoom);

CString m\_name;

CString m\_id;

int m\_left;

int m\_top;

int m\_right;

int m\_bottom;

//100%时左上角坐标

int m\_x;

int m\_y;

CStation \* m\_pStation;

vector<CBase \*> m\_graphs;

// 判断是否选中

bool isInRect(CPoint point);

// 复制

CPSDStationGraph \* clone(CDC \*pDC);

//实际的左上角坐标

int m\_realX;

int m\_realY;

int m\_realLeft;

int m\_realRight;

int m\_realTop;

int m\_realBottom;

// 场站颜色

COLORREF m\_colorref;

// 场站线条粗细

int m\_nLineWidth;

float getMaxBufKV();

};

#include "stdafx.h"

#include "PSDStationGraph.h"

CPSDStationGraph::CPSDStationGraph()

: m\_name(\_T(""))

, m\_id(\_T(""))

, m\_left(0)

, m\_top(0)

, m\_right(0)

, m\_bottom(0)

, m\_x(0)

, m\_y(0)

, m\_pStation(NULL)

, m\_realX(0)

, m\_realY(0)

, m\_nLineWidth(2)

{

m\_realLeft = 0;

m\_realRight = 0;

m\_realTop = 0;

m\_realBottom = 0;

m\_colorref = RGB(0, 128, 64);

}

CPSDStationGraph::~CPSDStationGraph()

{

}

void CPSDStationGraph::draw(CDC \* pDC, int zoom, int x, int y)

{

//if (m\_pStation->m\_isVirtualPoint) {

// m\_x = x;

// m\_y = y;

// m\_realX = x \* zoom / 100.0f;

// m\_realY = y \* zoom / 100.0f;

// m\_left = -3;

// m\_right = 3;

// m\_top = -3;

// m\_bottom = 3;

// m\_realLeft = m\_left \* zoom / 100.0f;

// m\_realRight = m\_right \* zoom / 100.0f;

// m\_realTop = m\_top \* zoom / 100.0f;

// m\_realBottom = m\_bottom \* zoom / 100.0f;

// int centerX = x + (m\_right - m\_left) / 2;

// int centerY = y + (m\_bottom - m\_top) / 2;

// centerX = centerX \* zoom / 100.0f;

// centerY = centerY \* zoom / 100.0f;

// CPoint point = pDC->SetViewportOrg(centerX, centerY);

// CPen pen(PS\_SOLID, 2, RGB(0, 0, 0));

// CPen \*pOldPen = pDC->SelectObject(&pen);

// int dpiX = pDC->GetDeviceCaps(LOGPIXELSX);

// int dpiY = pDC->GetDeviceCaps(LOGPIXELSY);

// int left = m\_left; /// 25.4f \* dpiX;

// int top = m\_top; // / 25.4f \* dpiY;

// int right = m\_right; // / 25.4f \* dpiX;

// int bottom = m\_bottom;// / 25.4f \* dpiY;

// pDC->Ellipse(left\*zoom / 100.0f, top\*zoom / 100.0f, right\*zoom / 100.0f, bottom\*zoom / 100.0f);

// pDC->SelectObject(pOldPen);

// pDC->SetViewportOrg(point);

// return;

//}

m\_x = x;

m\_y = y;

m\_realX = x \* zoom / 100.0f;

m\_realY = y \* zoom / 100.0f;

int width = m\_right - m\_left;

int height = m\_bottom - m\_top;

int centerX = x + width / 2;

int centerY = y + height / 2;

centerX = centerX \* zoom / 100.0f;

centerY = centerY \* zoom / 100.0f;

//width = width \* zoom / 100.0f;

//height = height \* zoom / 100.0f;

m\_realLeft = m\_left \* zoom / 100.0f;

m\_realRight = m\_right \* zoom / 100.0f;

m\_realTop = m\_top \* zoom / 100.0f;

m\_realBottom = m\_bottom \* zoom / 100.0f;

float baseKV = getMaxBufKV();

CPoint point = pDC->SetViewportOrg(centerX, centerY);

//pDC->BeginPath();

for (int i = 0; i < m\_graphs.size();i++) {

CBase \*p = m\_graphs[i];

/\*if ((m\_pStation->m\_buses.size() == 0) && (!m\_pStation->m\_isVirtualPoint)) {

p->m\_color = RGB(128, 128, 128);

}

else {

p->m\_color = m\_colorref;

}\*/

p->m\_color = m\_colorref;

if (m\_pStation->m\_isSelected) {

if (m\_colorref == RGB(0, 0, 0)) {

p->m\_color = RGB(255, 0, 0);

}

else {

p->m\_color = RGB(255, 255, 255) - m\_colorref;

}

}

p->m\_lineWidth = 2;

p->draw(pDC, zoom);

}

// pDC->EndPath();

// pDC->StrokePath();

/\*if (m\_pStation->m\_isSelected) {

CPen pen(PS\_DASH, 1, RGB(255, 0, 0));

CPen \*pOldPen = pDC->SelectObject(&pen);

int bkModel = pDC->SetBkMode(TRANSPARENT);

pDC->Rectangle(m\_realLeft - 3, m\_realTop - 3, m\_realRight + 3, m\_realBottom + 3);

pDC->SetBkMode(bkModel);

pDC->SelectObject(pOldPen);

pen.DeleteObject();

}\*/

pDC->SetViewportOrg(point);

float bus\_kv = 0.0f;

float pu = 0.0f;

CString kv = L"";

if (m\_pStation->m\_pDoc->m\_isDisplayStationData) {

if (m\_pStation->m\_voltPos < 0) {

m\_pStation->m\_voltPos = 0;

int i = 0;

if (m\_pStation->m\_buses.size() > 0) {

for (vector<CPSSEBus \*>::iterator it = m\_pStation->m\_buses.begin();

it != m\_pStation->m\_buses.end(); it++) {

if ((\*it)->volt\_KV > bus\_kv) {

bus\_kv = (\*it)->volt\_KV;

pu = (\*it)->volt\_PU;

m\_pStation->m\_voltPos = i;

}

i++;

}

}

}

else {

if (m\_pStation->m\_buses.size() > 0) {

bus\_kv = m\_pStation->m\_buses[m\_pStation->m\_voltPos]->volt\_KV;

pu = m\_pStation->m\_buses[m\_pStation->m\_voltPos]->volt\_PU;

}

else {

bus\_kv = 0;

pu = 0;

}

}

}

if (!(m\_pStation->m\_isVirtualPoint) && (m\_pStation->m\_pDoc->m\_isDisplayStationData)) {

if (m\_pStation->m\_isDisplayPowerflow && (m\_pStation->m\_buses.size() > 0)) {

float pa = 0.0f;

CPSSEBus \*p = m\_pStation->m\_buses[m\_pStation->m\_voltPos];

for (int i = 0; i < p->m\_branches.size(); i++) {

CPSSEBranch \*pBranche = p->m\_branches[i];

if ((pBranche->m\_type == 2)) {

CTransformerData \*pTransformerData = static\_cast<CTransformerData \*>(pBranche);

if (p == pTransformerData->m\_busPoints[0]) {

pa += pTransformerData->m\_IBus\_MW;

}

else if (p == pTransformerData->m\_busPoints[1]) {

pa += pTransformerData->m\_JBUS\_MW;

}

}

else if (pBranche->m\_type == 3) {

CTransformerData \*pTransformerData = static\_cast<CTransformerData \*>(pBranche);

if (p == pTransformerData->m\_busPoints[0]) {

pa += pTransformerData->IPAC;

}

else if (p == pTransformerData->m\_busPoints[1]) {

pa += pTransformerData->JPAC;

}

else {

pa += pTransformerData->KPAC;

}

}

}

CString temp1;

temp1.Format(L"P↓%0.\*f\n", m\_pStation->m\_pDoc->m\_nVolDec, pa);

CString temp2;

if (m\_pStation->m\_pDoc->m\_nDisplayUnit == 0) {

temp2.Format(L"%.\*f", m\_pStation->m\_pDoc->m\_nVolDec, bus\_kv);

}

else {

temp2.Format(L"%.\*f", m\_pStation->m\_pDoc->m\_nPuVoltDec, pu);

}

kv = temp1 + temp2;

//kv.Format(L"P↓%0.2f\n%.1f", pa, bus\_kv);

/\*if (pa > 0.0f) {

kv.Format(L"P↓%0.2f\n%.1f", pa, bus\_kv);

}

else {

kv.Format(L"P↑%0.2f\n%.1f", fabs(pa), bus\_kv);

}\*/

}

else {

if (m\_pStation->m\_pDoc->m\_nDisplayUnit == 0) {

kv.Format(L"%.\*f", m\_pStation->m\_pDoc->m\_nVolDec, bus\_kv);

}

else {

kv.Format(L"%.\*f", m\_pStation->m\_pDoc->m\_nPuVoltDec, pu);

}

}

m\_pStation->m\_pVoltLabel->text = kv;

m\_pStation->m\_pVoltLabel->draw(pDC, zoom);

}

}

void CPSDStationGraph::drawSelectedRect(CDC \* pDC,int zoom) {

int width = m\_right - m\_left;

int height = m\_bottom - m\_top;

CRect rect;

rect.bottom = m\_y + height;

rect.top = m\_y;

rect.left = m\_x;

rect.right = m\_x + width;

CRect rect1(0, 0, 0, 0);

pDC->Rectangle(rect);

}

// 判断是否选中

bool CPSDStationGraph::isInRect(CPoint point)

{

int width = m\_realRight - m\_realLeft;

int height = m\_realBottom - m\_realTop;

CRect rect;

rect.bottom = m\_realY + height;

rect.top = m\_realY;

rect.left = m\_realX;

rect.right = m\_realX + width;

return PtInRect(&rect, point);

}

// 复制

CPSDStationGraph \* CPSDStationGraph::clone(CDC \*pDC)

{

int dpiX = pDC->GetDeviceCaps(LOGPIXELSX);

int dpiY = pDC->GetDeviceCaps(LOGPIXELSY);

CPSDStationGraph \*p = new CPSDStationGraph();

p->m\_bottom = m\_bottom / 96.0f \* dpiY;

p->m\_top = m\_top / 96.0f \* dpiY;

p->m\_left = m\_left / 96.0f \* dpiX;

p->m\_right = m\_right / 96.0f \* dpiX;

p->m\_id = m\_id;

p->m\_name = m\_name;

p->m\_x = m\_x;

p->m\_y = m\_y;

p->m\_graphs.assign(m\_graphs.begin(), m\_graphs.end());

p->m\_nLineWidth = m\_nLineWidth;

return p;

}

float CPSDStationGraph::getMaxBufKV()

{

float bus\_kv = 0;

if (m\_pStation->m\_buses.size() > 0) {

for (vector<CPSSEBus \*>::iterator it = m\_pStation->m\_buses.begin();

it != m\_pStation->m\_buses.end(); it++) {

if ((\*it)->volt\_KV > bus\_kv) {

bus\_kv = (\*it)->volt\_KV;

}

}

}

return bus\_kv;

}

#pragma once

#include "PSSEBranch.h"

class CNonTransformerBranchData :

public CPSSEBranch

{

public:

CNonTransformerBranchData();

~CNonTransformerBranchData();

CString getIdenetifyName();

// unsigned int I;

// unsigned int J;

CString I;

CString J;

CString CKT;

float R;

float X;

float B;

float GI;

float BI;

float GJ;

float BJ;

int STAT;

int MET;

float LEN;

};

#include "stdafx.h"

#include "NonTransformerBranchData.h"

CNonTransformerBranchData::CNonTransformerBranchData()

{

// I = 0;

// J = 0;

I = \_T("");

J = \_T("");

CKT = \_T("");

R = 0.0f;

X = 0.0f;

B = 0.0f;

GI = 0.0f;

BI = 0.0f;

GJ = 0.0f;

BJ = 0.0f;

STAT = 0;

MET = 0;

LEN = 0.0f;

}

CNonTransformerBranchData::~CNonTransformerBranchData()

{

}

CString CNonTransformerBranchData::getIdenetifyName() {

return I + J + CKT;

}

#include "stdafx.h"

#include "PSSEBus.h"

CPSSEBus::CPSSEBus()

: m\_uiID(0)

, m\_strName(\_T(""))

, m\_fBaseKV(0)

, m\_type(0)

, m\_area(0)

, m\_zone(0)

, m\_owner(0)

, pStation(NULL)

, m\_isDrawed(FALSE)

, m\_isSelected(FALSE)

, m\_isNeutralPoint(false)

{

volt\_PU=0.0f;

volt\_KV=0.0f;

angle = 0.0f;

gen\_mw = 0.0f;

gen\_mvar = 0.0f;

load\_mw = 0.0f;

load\_mvar = 0.0f;

shunt\_mw = 0.0f;

shunt\_mvar = 0.0f;

}

CPSSEBus::~CPSSEBus()

{

for (int i = 0; i < m\_loadDatas.size();i++) {

CLoadBus \* p = m\_loadDatas[i];

delete p;

p = NULL;

m\_loadDatas[i] = NULL;

}

m\_loadDatas.clear();

for (int i = 0; i < m\_fixedBusShuntDatas.size(); i++) {

CFixedBusShuntData \*p = m\_fixedBusShuntDatas[i];

delete p;

p = NULL;

m\_fixedBusShuntDatas[i] = NULL;

}

m\_fixedBusShuntDatas.clear();

for (int i = 0; i < m\_generatorDatas.size(); i++) {

CGeneratorData \*p = m\_generatorDatas[i];

delete p;

p = NULL;

m\_generatorDatas[i] = NULL;

}

m\_generatorDatas.clear();

}