# 通用单文档图像显示框架

这里总结一个通用单文档图像显示框架，另外也附上一些扩展的功能和常遇到的问题，以供后来参考。

该框架采用单文档分割视图的形式，左侧为From类型窗口，右侧为滚动视图。

## 1.基本实现

### 2.1.视图的分割

新建单文档工程，视图类基于CFormView类，另外再自己添加一个ImageView类，基于CScrollView类。

在MainFrm中添加一个CSplitterWnd m\_wndSplitter成员，并修改其OnCreateClient函数：

|  |
| --- |
| *BOOL* CMainFrame::OnCreateClient(*LPCREATESTRUCT* lpcs, *CCreateContext*\* pContext)  {  *CRect* rc;  *GetClientRect*(&rc);  if (!m\_wndSplitter.*CreateStatic*(this, 1, 2))  return *FALSE*;  if (!m\_wndSplitter.*CreateView*(0, 0, *RUNTIME\_CLASS*(CAGVCalibrationView),  *CSize*(300, rc.*Height*()), pContext))  return *FALSE*;  if (!m\_wndSplitter.*CreateView*(0, 1, *RUNTIME\_CLASS*(ImageView),  *CSize*(rc.*Width*() - 300, rc.*Height*()), pContext))  return *FALSE*;  return *TRUE*;  } |

### 3.2.视图的通信

为方便视图间通信，在各视图中添加一个获取另一个视图指针的函数，例如这里的ImageView中：

|  |
| --- |
| *CView*\* ImageView::GetPanelView()  {  return (*CView*\*)((CMainFrame\*)*AfxGetMainWnd*())->m\_wndSplitter.*GetPane*(0, 0);  } |

### 3.3.图像的显示

这里在Form视图中使用OpenCV读取和处理图像图像，在ImageView中使用ImgCenterDib类来显示图像。

在ImageView中维护以下几个成员：

|  |
| --- |
| void ShowImage(*cv*::*Mat* img, *BOOL* bErase);  void ClearScreen();  private:  ImgCenterDib m\_Dib;  *cv*::*Mat* img;  *cv*::*Mat* tmpImg; |

函数实现：

|  |
| --- |
| void ImageView::ShowImage(*cv*::*Mat* inputMat, *BOOL* bErase)  {  *CSize* sizeTotal;  img = inputMat.*clone*();  tmpImg = inputMat.*clone*();  // TODO: calculate the total size of this view  sizeTotal.*cx* = MAX(img.*cols*, 1000);  sizeTotal.*cy* = MAX(img.*rows*, 1000);;  *SetScrollSizes*(*MM\_TEXT*, sizeTotal);  *Invalidate*(bErase);  }  void ImageView::ClearScreen()  {  *ShowWindow*(*FALSE*);  img = *cv*::*Mat*();  tmpImg = *cv*::*Mat*();  *ShowWindow*(*TRUE*);  } |

同时，还要对OnDraw函数进行修改：

|  |
| --- |
| void ImageView::OnDraw(*CDC*\* pDC)  {  *CDocument*\* pDoc = GetDocument();  if (tmpImg.*data*)  {  m\_Dib.ConvertOpenCVMatToDIB(tmpImg, 1);  m\_Dib.Draw(pDC, *CPoint*(0, 0), m\_Dib.GetDimensions());  }  } |

在Form视图中进行图像处理后，利用视图通信机制，调用ImageView的ShowImage函数即可实现视图的显示和更新

### 3.4.滚动视图的使用

滚动视图与普通视图相比，增加了滚动条，这导致视图滚动后，设备坐标与视图坐标不再对应。这在用鼠标对图像进行操作时会受到影响，解决方法是通过DPtoLP函数进行转换，例如，下面是对鼠标拾取图像点的处理：

|  |
| --- |
| void ImageView::OnLButtonDown(*UINT* nFlags, *CPoint* point)  {  // TODO: Add your message handler code here and/or call default  *CPoint* pt = point;  *CDC*\* pDC = *GetDC*();  *OnPrepareDC*(pDC);  pDC->*DPtoLP*(&pt);  currentPt = *cv*::*Point*(pt.*x*, pt.*y*);  //将点增加到点集中  ((CAGVCalibrationView\*)GetPanelView())->addPoint(currentPt);  *ReleaseDC*(pDC);  *CScrollView*::*OnLButtonDown*(nFlags, point);  *Invalidate*(0);  } |

在对图像绘图时，应在OnDraw中实现，例如，这里将Form视图中维护的点集显示到视图中：

|  |
| --- |
| void ImageView::OnDraw(*CDC*\* pDC)  {  *CDocument*\* pDoc = GetDocument();  CAGVCalibrationView\* pPanelView = (CAGVCalibrationView\*)GetPanelView();  if (tmpImg.*data*)  {  *std*::*vector*<*std*::*pair*<*cv*::*Point*, int>> PointSet = pPanelView->getPointSet();  for (int i = 0; i < PointSet.*size*();i++)  {  *cv*::*Scalar* color = *cv*::*Scalar*(0, 0, 255);  *cv*::*circle*(tmpImg, PointSet[i].*first*, 3, color, -1);  }  m\_Dib.ConvertOpenCVMatToDIB(tmpImg, 1);  m\_Dib.Draw(pDC, *CPoint*(0, 0), m\_Dib.GetDimensions());  }  } |

## 2.扩展功能

### 2.1 Tab控件的使用

在Form类中维护以下几个变量：

|  |
| --- |
| public:  *BOOL* bCreate;  *CTabCtrl* m\_ctrTab;  CFishCalibrationDlg m\_fishCalibrationDlg;  CHeightCalibrationDlg m\_heightCalibrationDlg;  CTransformDlg m\_transformDlg;  CSummaryDlg m\_summaryDlg; |

修改OnInitialUpdate文件：

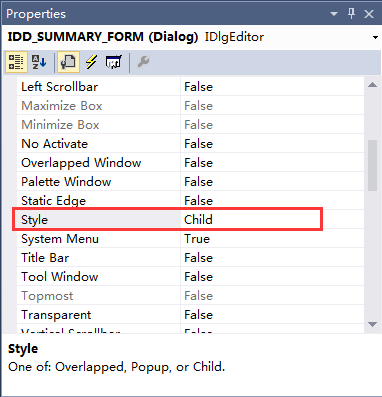
|  |
| --- |
| void CAGVCalibrationView::OnInitialUpdate()  {  *CFormView*::*OnInitialUpdate*();  *GetParentFrame*()->*RecalcLayout*();  *ResizeParentToFit*();  if (!bCreate)  {  //初始化Tab窗口  m\_fishCalibrationDlg.*Create*(IDD\_FISH\_CALIBRATION\_DLG);  m\_fishCalibrationDlg.*SetParent*(&m\_ctrTab);  m\_fishCalibrationDlg.pMainPanel = this;  m\_fishCalibrationDlg.fScale = fScale;  m\_heightCalibrationDlg.*Create*(IDD\_HEIGHT\_CALIBRATION\_DLG);  m\_heightCalibrationDlg.*SetParent*(&m\_ctrTab);  m\_heightCalibrationDlg.pMainPanel = this;  m\_heightCalibrationDlg.fScale = fScale;  m\_transformDlg.*Create*(IDD\_DLG\_TRANSFORM);  m\_transformDlg.*SetParent*(&m\_ctrTab);  m\_transformDlg.pMainPanel = this;  m\_transformDlg.fScale = fScale;  m\_summaryDlg.*Create*(IDD\_DLG\_SUMMARY);  m\_summaryDlg.*SetParent*(&m\_ctrTab);  m\_ctrTab.*InsertItem*(0, *\_T*("鱼眼校正"));  m\_ctrTab.*InsertItem*(1, *\_T*("高度标定"));  m\_ctrTab.*InsertItem*(2, *\_T*("坐标转换"));  m\_ctrTab.*InsertItem*(3, *\_T*("汇总"));  *CRect* rect;  m\_ctrTab.*GetClientRect*(rect);  //适当的减小客户区  rect.*DeflateRect*(5, 25, 5, 5);  m\_fishCalibrationDlg.*MoveWindow*(&rect);  m\_fishCalibrationDlg.*ShowWindow*(*SW\_SHOW*);  m\_heightCalibrationDlg.*MoveWindow*(&rect, *TRUE*);  m\_heightCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_transformDlg.*MoveWindow*(&rect, *TRUE*);  m\_transformDlg.*ShowWindow*(*SW\_HIDE*);  m\_summaryDlg.*MoveWindow*(&rect, *TRUE*);  m\_summaryDlg.*ShowWindow*(*SW\_HIDE*);  m\_ctrTab.*SetCurSel*(0);  bCreate = *TRUE*;  }  } |

在Tab控件选项变化的时候，执行相应的动作：

|  |
| --- |
| void CAGVCalibrationView::OnTcnSelchangeTabItems(*NMHDR* \*pNMHDR, *LRESULT* \*pResult)  {  // TODO: Add your control notification handler code here  int index = m\_ctrTab.*GetCurSel*();  CMainFrame\* pFrame = (CMainFrame\*)*AfxGetMainWnd*();  switch (index)  {  case 0:  pFrame->SwitchView(IMAGE\_VIEW);  m\_fishCalibrationDlg.*ShowWindow*(*SW\_SHOW*);  m\_heightCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_transformDlg.*ShowWindow*(*SW\_HIDE*);  break;  case 1:  pFrame->SwitchView(IMAGE\_VIEW);  m\_fishCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_heightCalibrationDlg.*ShowWindow*(*SW\_SHOW*);  m\_transformDlg.*ShowWindow*(*SW\_HIDE*);  break;  case 2:  pFrame->SwitchView(IMAGE\_VIEW);  m\_fishCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_heightCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_transformDlg.*ShowWindow*(*SW\_SHOW*);  break;  case 3:  pFrame->SwitchView(SUMMARY\_VIEW);  m\_fishCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_heightCalibrationDlg.*ShowWindow*(*SW\_HIDE*);  m\_transformDlg.*ShowWindow*(*SW\_HIDE*);  m\_summaryDlg.*ShowWindow*(*SW\_SHOW*);  break;  }  FlushImageView();  \*pResult = 0;  } |

### 2.2 视图的切换

这里目的是实现ImageView和另一个视图相互切换，插入一个对话框，Border属性选择None，Style属性选择Child



创建一个基于该对话框，继承自CFormView类，名为CSummaryFormView类。在CMainFrm中实现SwitchView函数：

|  |
| --- |
| void CMainFrame::SwitchView(int iViewType)  {  *CView*\* pView = (*CView*\*)m\_wndSplitter.*GetPane*(0, 1);  *CRect* rcRight, rcFrame;  pView->*GetClientRect*(&rcRight);  *GetClientRect*(&rcFrame);  switch (iViewType)  {  case IMAGE\_VIEW:  if (!pView->*IsKindOf*(*RUNTIME\_CLASS*(ImageView)))  {  m\_wndSplitter.*DeleteView*(0, 1);  m\_wndSplitter.*CreateView*(0, 1, *RUNTIME\_CLASS*(ImageView),  *CSize*(rcRight.*Width*(), rcRight.*Height*()), *NULL*);  m\_wndSplitter.*RecalcLayout*();  }  break;  case SUMMARY\_VIEW:  if (!pView->*IsKindOf*(*RUNTIME\_CLASS*(CSummaryFormView)))  {  m\_wndSplitter.*DeleteView*(0, 1);  m\_wndSplitter.*CreateView*(0, 1, *RUNTIME\_CLASS*(CSummaryFormView),  *CSize*(rcRight.*Width*(), rcRight.*Height*()), *NULL*);  m\_wndSplitter.*RecalcLayout*();  }  break;  default:  break;  }  } |

其中IMAGE\_VIEW和SUMMARY\_VIEW是自定义的宏，在需要转换视图的地方，调用该函数即可。

### 2.3 目录选择对话框的使用

下面的函数演示了目录选择对话框的使用：

|  |
| --- |
| static int *CALLBACK* BrowseCallbackProc(*HWND* hwnd, *UINT* uMsg, *LPARAM* lParam, *LPARAM* lpData)  {  switch (uMsg)  {  case *BFFM\_INITIALIZED*: //初始化消息  ::*SendMessage*(hwnd, *BFFM\_SETSELECTION*, *TRUE*,  (*LPARAM*)(*LPTSTR*)(*LPCTSTR*)strWorkDir);  break;  case *BFFM\_SELCHANGED*: //选择路径变化，  {  *TCHAR* curr[*MAX\_PATH*];  *SHGetPathFromIDList*((*LPCITEMIDLIST*)lParam, curr);  ::*SendMessage*(hwnd, *BFFM\_SETSTATUSTEXT*, 0, (*LPARAM*)curr);  }  break;  default:  break;  }  return 0;  }  void CFishCalibrationDlg::OnBnClickedBtnOpenFish()  {  *BROWSEINFO* bi;  *WCHAR* Buffer[512];  bi.*hwndOwner* = *NULL*;  bi.*pidlRoot* = *NULL*;  bi.*pszDisplayName* = Buffer;  bi.*lpszTitle* = *\_T*("选择目录");  bi.*ulFlags* = *BIF\_RETURNONLYFSDIRS*;  bi.*lpfn* = *NULL*;  bi.lParam = long(&strWorkDir);  strWorkDir.*ReleaseBuffer*();  bi.*iImage* = 0;  bi.*lpfn* = BrowseCallbackProc;  *LPITEMIDLIST* pIDList = *SHBrowseForFolder*(&bi);  if (pIDList)  {  *SHGetPathFromIDList*(pIDList, Buffer);  CString strImageDir = Buffer;  //对文件目录进行处理  }  } |

以上strWorkDir是CString类型变量，为根目录的路径。

### 2.4 软件的加密

添加一个名为CEncryptor的类，实现如下：

Encryptor.h:

|  |
| --- |
| #pragma once  class CEncryptor  {  public:  CEncryptor();  ~CEncryptor();  *CString* Encrypt(*CString* S, *WORD* Key);  *CString* Decrypt(*CString* S, *WORD* Key);  *CString* GetMechineCode();  }; |

Encryptor.cpp

|  |
| --- |
| #include "stdafx.h"  #include "Encryptor.h"  #define C1 52845  #define C2 22719  CEncryptor::CEncryptor()  {  }  CEncryptor::~CEncryptor()  {  }  *CString* CEncryptor::Encrypt(*CString* S, *WORD* Key) // 加密函数  {  *CString* Result, str;  int i, j;  Result = S; // 初始化结果字符串  for (i = 0; i < S.*GetLength*(); i++) // 依次对字符串中各字符进行操作  {  Result.*SetAt*(i, S.*GetAt*(i) ^ (Key >> 8)); // 将密钥移位后与字符异或  Key = ((*BYTE*)Result.*GetAt*(i) + Key)\*C1 + C2; // 产生下一个密钥  }  S = Result; // 保存结果  Result.*Empty*(); // 清除结果  for (i = 0; i < S.*GetLength*(); i++) // 对加密结果进行转换  {  j = (*BYTE*)S.*GetAt*(i); // 提取字符  // 将字符转换为两个字母保存  str = "12"; // 设置str长度为2  str.*SetAt*(0, 65 + j / 26);//这里将65改大点的数例如256，密文就会变乱码，效果更好，相应的，解密处要改为相同的数  str.*SetAt*(1, 65 + j % 26);  Result += str;  }  return Result;  }  // 解密函数  *CString* CEncryptor::Decrypt(*CString* S, *WORD* Key)  {  *CString* Result, str;  int i, j;  Result.*Empty*(); // 清除结果  for (i = 0; i < S.*GetLength*() / 2; i++) // 将字符串两个字母一组进行处理  {  j = ((*BYTE*)S.*GetAt*(2 \* i) - 65) \* 26;//相应的，解密处要改为相同的数  j += (*BYTE*)S.*GetAt*(2 \* i + 1) - 65;  str = "1"; // 设置str长度为1  str.*SetAt*(0, j);  Result += str; // 追加字符，还原字符串  }  S = Result; // 保存中间结果  for (i = 0; i < S.*GetLength*(); i++) // 依次对字符串中各字符进行操作  {  Result.*SetAt*(i, (*BYTE*)S.*GetAt*(i) ^ (Key >> 8)); // 将密钥移位后与字符异或  Key = ((*BYTE*)S.*GetAt*(i) + Key)\*C1 + C2; // 产生下一个密钥  }  return Result;  }  *CString* CEncryptor::GetMechineCode()  {  *TCHAR* m\_Volume[256];//卷标名  *TCHAR* m\_FileSysName[256];  *DWORD* m\_SerialNum;//序列号  *DWORD* m\_FileNameLength;  *DWORD* m\_FileSysFlag;  ::*GetVolumeInformation*(*\_T*("c:\\"),  m\_Volume,  256,  &m\_SerialNum,  &m\_FileNameLength,  &m\_FileSysFlag,  m\_FileSysName,  256);  *CString* Serial;  Serial.*Format*(*\_T*("1%04x"), m\_SerialNum ^ 0x34795814);  return Serial;  } |

新建一个工程来生成License文件：

|  |
| --- |
| void CLicenseGeneratorDlg::OnBnClickedBtnGen()  {  // TODO: Add your control notification handler code here  *UpdateData*(*TRUE*);  if (m\_strMechineCode.*GetLength*()!=9)  {  *MessageBox*(*\_T*("请输入9位机器码"), *\_T*("提示"), *MB\_OK*);  *SetDlgItemText*(IDC\_EDIT1, *\_T*(""));  return;  }  *WORD* key = 1234;  *CString* filter = *\_T*("License File|\*.lic||");  *CFileDialog* dlg(*FALSE*, *\_T*(".lic"), *NULL*, *OFN\_CREATEPROMPT* | *OFN\_PATHMUSTEXIST*,  filter);  *CString* defaultName = *\_T*("license.lic");  dlg.*m\_ofn*.*lpstrFile* = defaultName.*GetBuffer*(*MAX\_PATH*);  if (dlg.*DoModal*() == *IDOK*)  {  CEncryptor entryptor;  *std*::*string* strResult = CString2String(entryptor.Encrypt(m\_strMechineCode, key));  *std*::*ofstream* file;  *std*::*string* filename = CString2String(dlg.*GetPathName*());  file.*open*(filename.*c\_str*());  if (!file.*is\_open*())  {  *MessageBox*(*\_T*("打开文件错误！"), *\_T*("提示"), *MB\_OK*);  file.*close*();  return;  }  file << strResult;  file.*close*();  }  defaultName.*ReleaseBuffer*();  } |

在MainFrm.cpp中修改

|  |
| --- |
| *BOOL* CMainFrame::PreCreateWindow(*CREATESTRUCT*& cs)  {  if( !*CFrameWnd*::*PreCreateWindow*(cs) )  return *FALSE*;  // TODO: Modify the Window class or styles here by modifying  // the CREATESTRUCT cs  cs.*style* &= ~*FWS\_ADDTOTITLE*;  CEncryptor entrytor;  *CString* mechineCode = entrytor.GetMechineCode();  if (mechineCode.*GetLength*()>9)  {  mechineCode = mechineCode.*Left*(9);  }  *WORD* key = 1234;  *CString* strCiphertext = entrytor.Encrypt(mechineCode, key);  *CString* strModulePath = GetModulePath();  *CString* strListense = LoadLicense(strModulePath + *\_T*("license\\license.lic"));  if (strListense!=strCiphertext)  {  ::*MessageBox*(*NULL*, *\_T*("软件尚未注册！您的机器码是：") + mechineCode  + *\_T*("，请联系软件商获取许可"), *\_T*("提示"), *MB\_OK*|*MB\_ICONINFORMATION*);  *exit*(-1);  }  return *TRUE*;  } |

其中LoadLicense函数的实现如下：

|  |
| --- |
| *CString* LoadLicense(*CString* strPath)  {  *CString* strResult;  *CStdioFile* File;  *BOOL* bRes = File.*Open*(strPath, *CFile*::*modeRead*);  if (!bRes)  {  ::*MessageBox*(*NULL*, *\_T*("未找到许可文件"), *\_T*("提示"),  *MB\_OK*|*MB\_ICONINFORMATION*);  *exit*(-1);  }  else  {  File.*ReadString*(strResult);  }  File.*Close*();  return strResult;  } |

## 3.其它问题

### 3.1 CString 和std::string的转换

|  |
| --- |
| *std*::*string* CString2String(*CString* str)  {  *USES\_CONVERSION*;  *std*::*string* str1;  str1 = *W2A*(str);  return str1;  } |

### 3.2 从文件路径中获取文件名

|  |
| --- |
| *CString* GetFileName(*CString* filePath, *BOOL* isWithFileSuffix)  {  int pos = filePath.*ReverseFind*('\\');  *CString* fileName = filePath.*Right*(filePath.*GetLength*() - pos - 1);  if (*FALSE* == isWithFileSuffix)  {  pos = fileName.*Find*('.');  fileName = fileName.*Left*(pos);  }  return fileName;  } |