**版本信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本信息记录表** | | | | |
| 版本号 | 修订内容 | 修订人 | 修订日期 | 审核人 |
| 1.0 | 初版 | 党慧杰 | 2018.08.20 |  |
| 2.0 | 修订 | 党慧杰 | 2018.10.09 |  |
|  |  |  |  |  |

# 面部特征识别

## 【实验目的】

1、利用Kinect采集彩色图像和骨骼图像，并进行面部特征的提取。

## 【实验原理】

Kinect获取骨骼图像，利用kinect的脸部特征提取算法，提取出识别到的人脸脸部特征。

## 【实验环境】

**开发环境：**

Windows 8或Windows 10系统、VS2012及以上版本、 Kinect设备、PC机必须具备USB3.0接口。

**编程语言:**

C/C++

## 【实验步骤】

### 一、打开项目

先将文档中的【KinectFaceFeatures】文件夹拷贝到一个无中文路径的目录下。注意：路径中一定要无中文，否则运行会出现错误。

打开“Visual Studio”。点击“文件/打开/项目/解决方案”，选择文档目录下的工程“KinectFaceFeatures”，如图中所示。

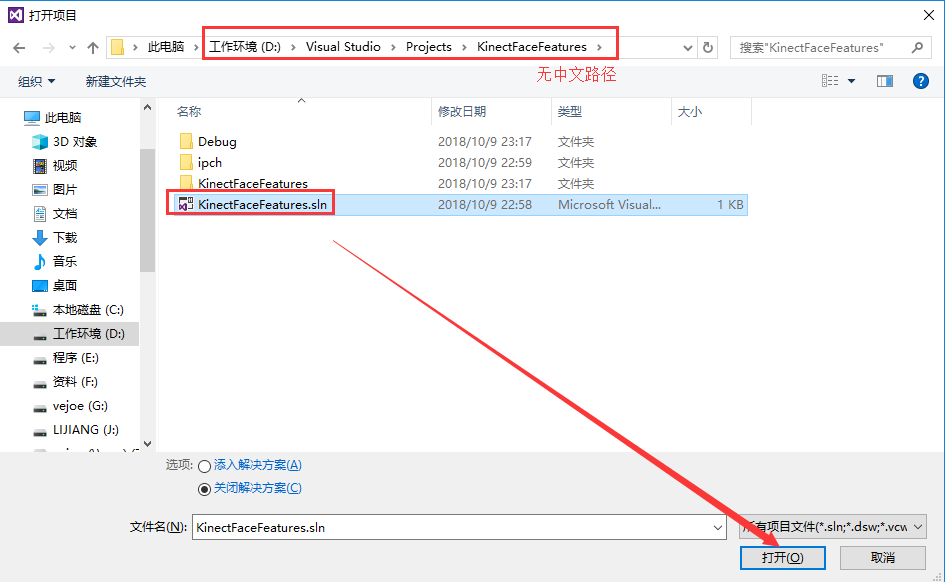


图 1

点击“打开”。

### 二、配置环境

1. 添加依赖库，鼠标右击项目->属性。在弹出的属性页中输入->附加依赖项中添加Kinect20.Face.lib和Dwrite.lib。如下图所示：

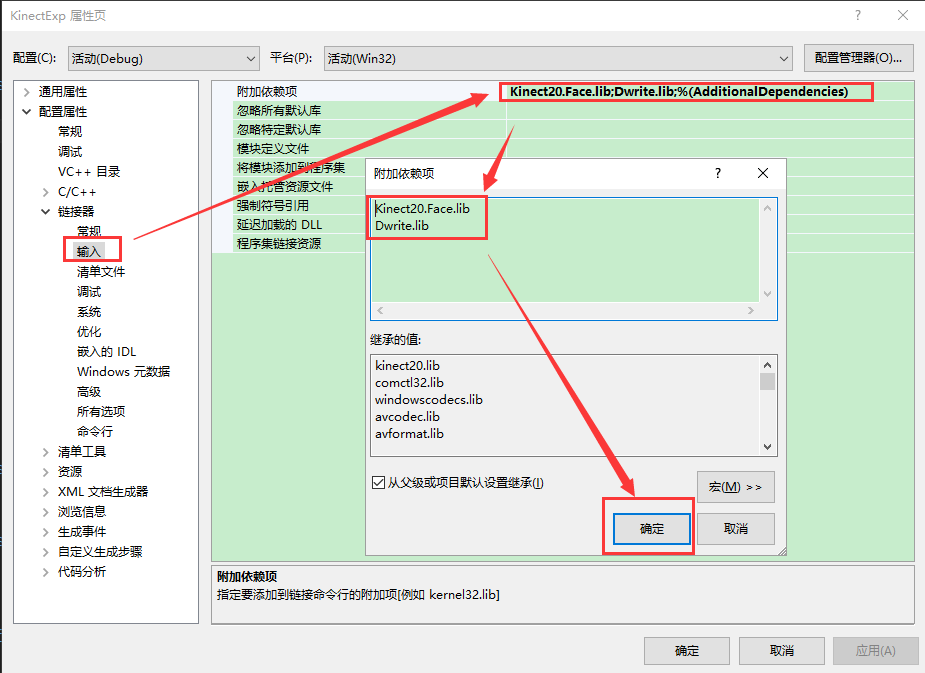


图 3

2. 将文档目录【faceLib】文件夹中的【NuiDatabase’】文件夹和Kinect20.Face.dll拷贝到与KinectFaceFeatures.exe同级目录下（先编译运行一下程序），如下图所示:



图4

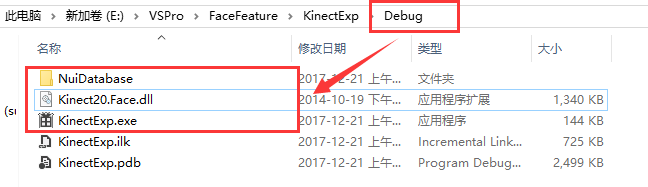


图5

### 三、在界面添加按钮

在界面添加一个按钮，当点击这个按钮时跳转到Kinect面部特征识别子窗口。

1）在KinectFaceFeatures.cpp中找到函数CreateMainWindow(HWND)。添加如下代码：

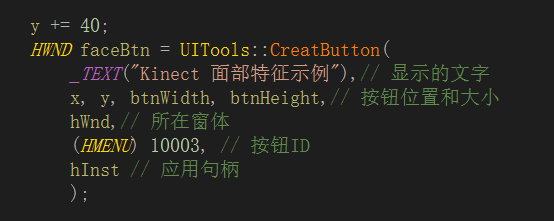


图6

2）响应按钮的点击事件

在主窗口的消息处理函数WndProc的switch(wmId)代码块中添加按钮点击事件响应，如下图所示：

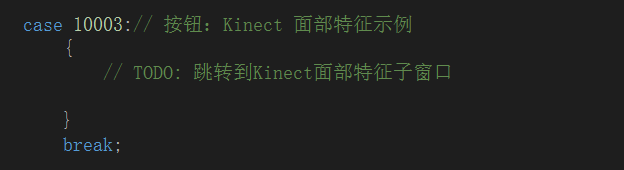


图7

可以编译运行，查看效果。

### 四、创建Kinect面部特征子窗口

点击“解决方案资源管理器”底部的资源视图，找到“Dialog”，右击鼠标，选择“添加资源”，在弹出的框中选择“Dialog”，点击“新建”。如下图所示：

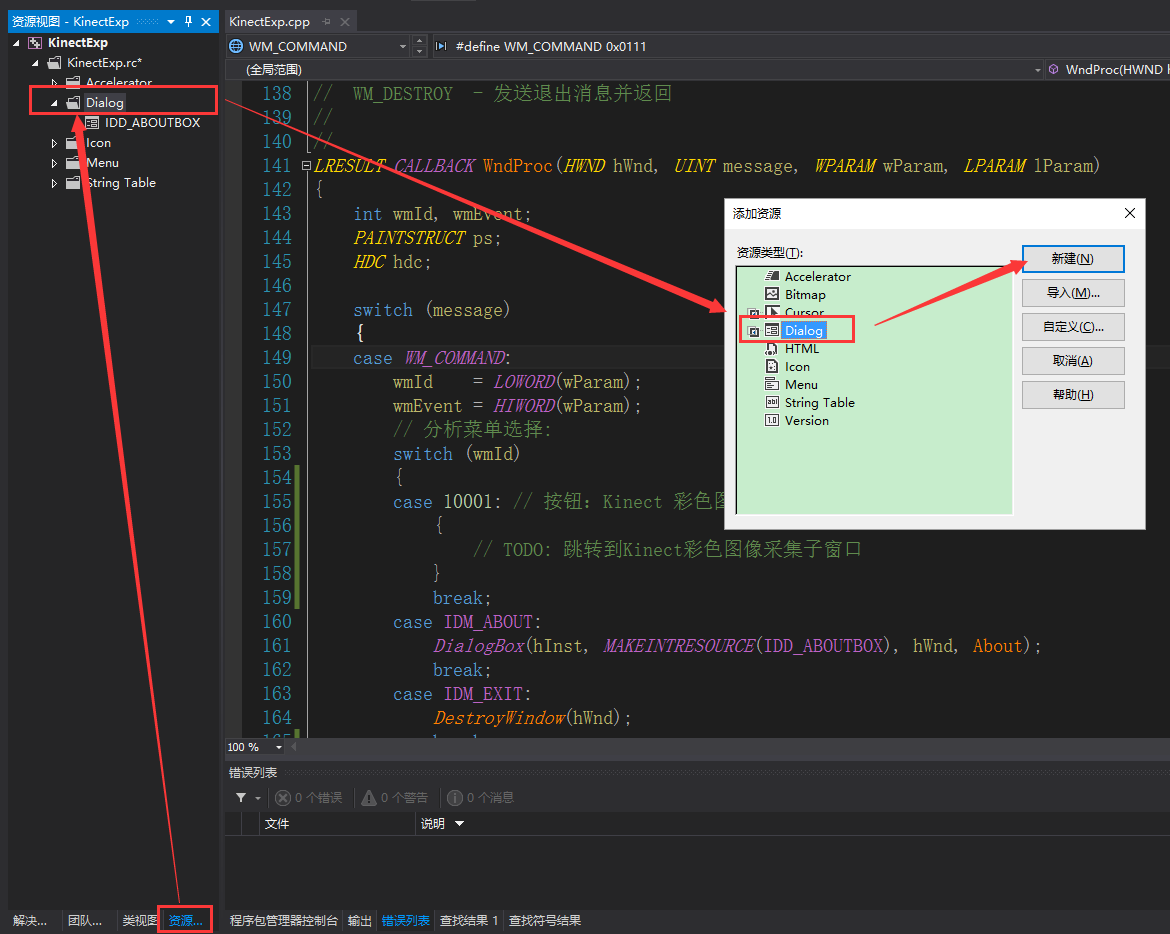


图8

将窗体，拖动到适当大小（640x374）。在左侧的属性栏中，将ID修改为：IDD\_FACE，Caption修改为：Kinect 面部特征，ClassName修改为：FaceDlgWndClass。如下图所示：

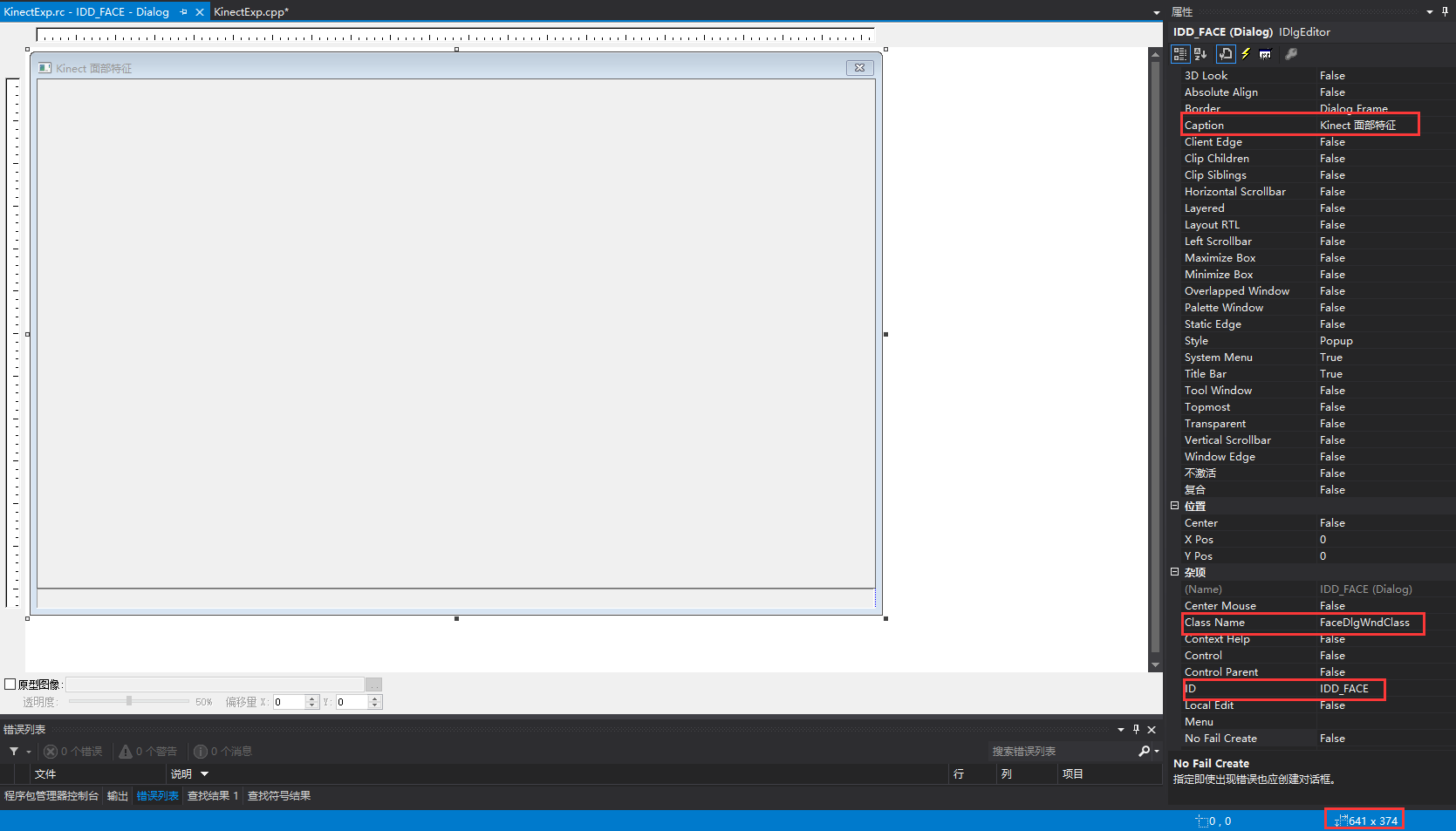


图9

在窗口中添加一个Picture Control控件，点击“工具箱”，找到Picture Control控件，并将其拖至窗口中合适的位置，适当调整大小，并设置如下的属性：将ID设置为：IDC\_FACE如下图所示。

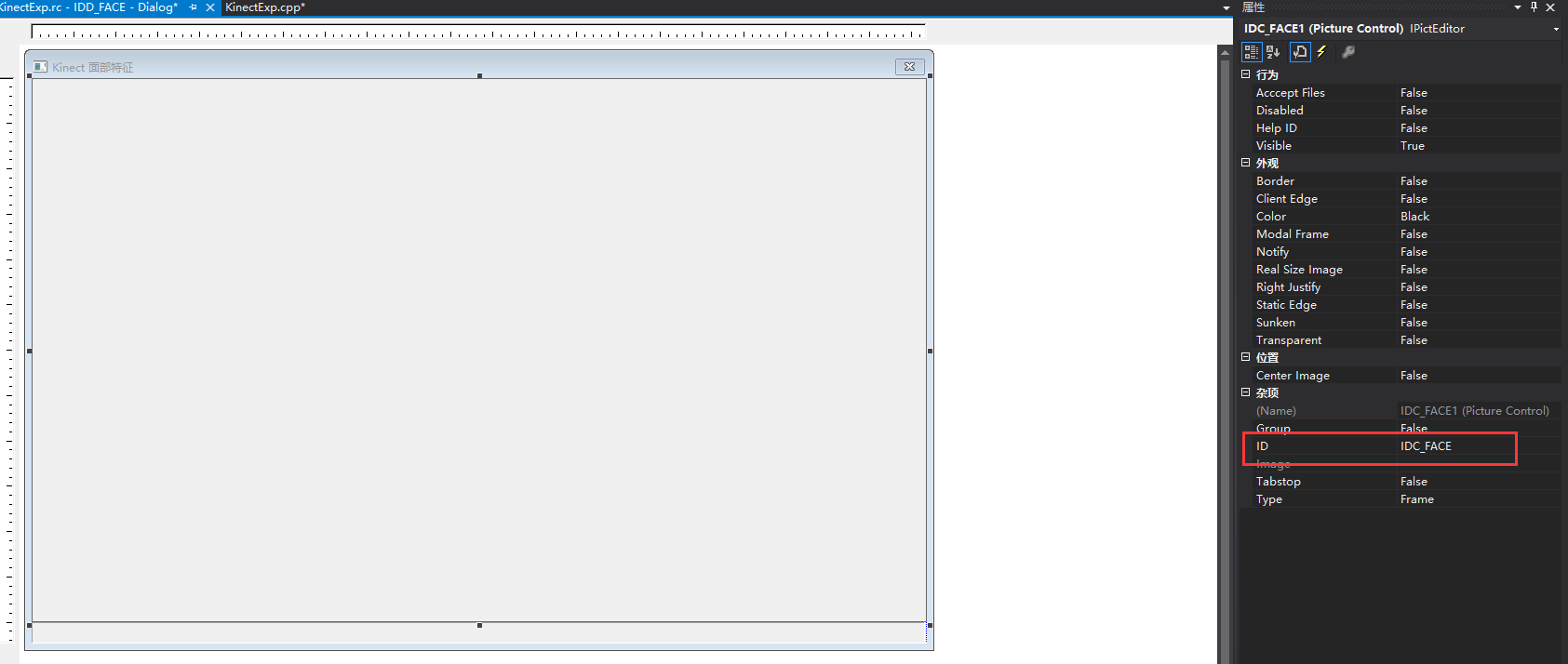


图10

用同样的方式添加静态文本(Static Text)，ID设置为：IDC\_KinectFaceStatus将Sunken属性设置为：True，如下图所示：

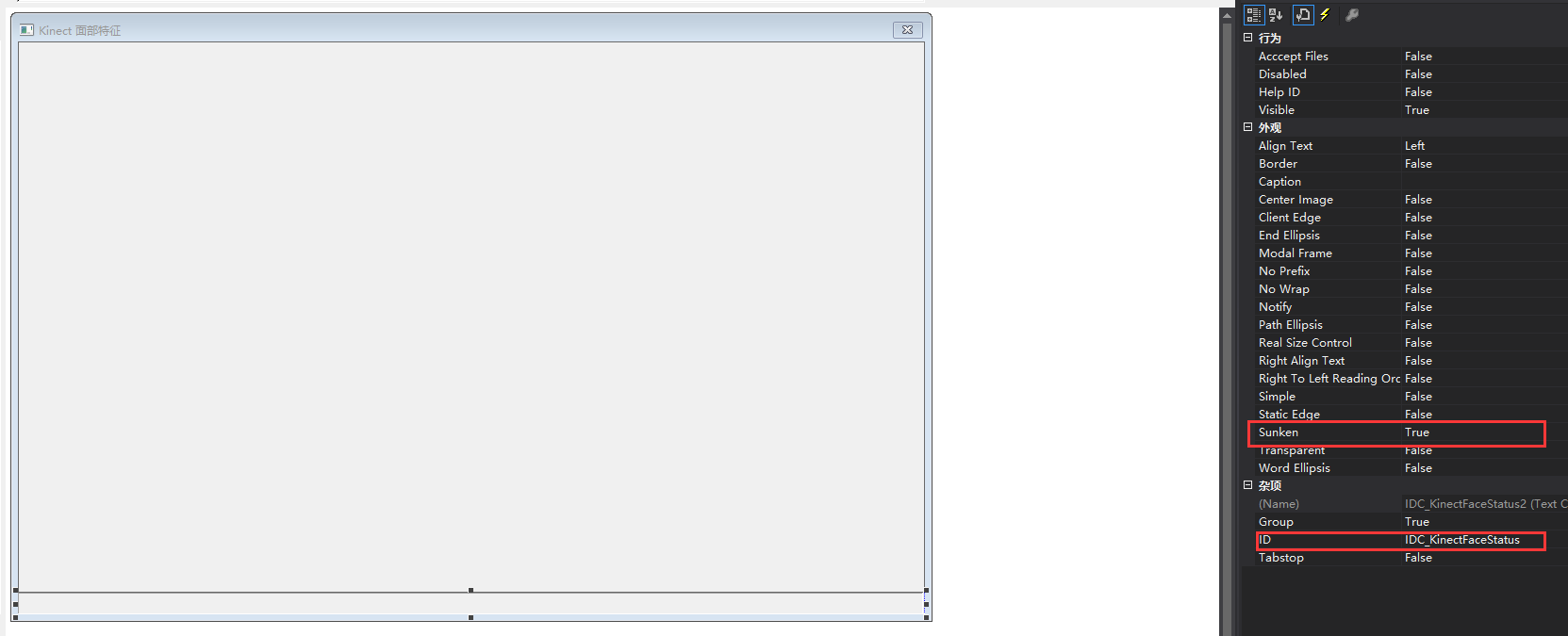


图11

### 五、创建一个新类KinectFaceBasics作为子窗口的载体

创建方法：鼠标右击解决方案，选择“添加”，选择“类”，在添加类的对话框中选择“C++类”，点击“创建”出现类创建向导，键入类名为KinectFaceBasics，如下图所示。



图12

**1．声明函数和变量**

在KinectFaceBasics.h中添加KinectDevice.h头文件如下所示：

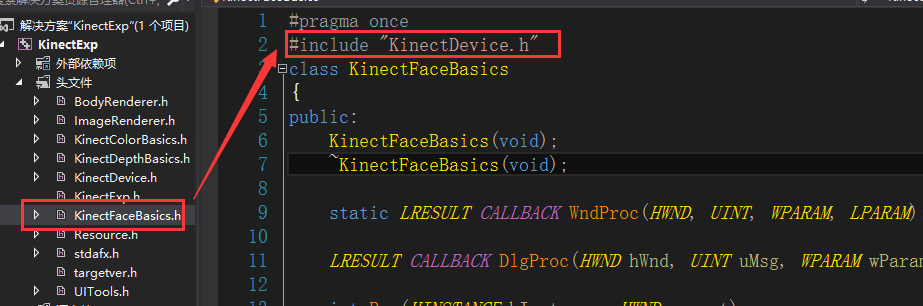


图13

同时在KinectFaceBasics.h中声明必要的函数和变量如下图所示：

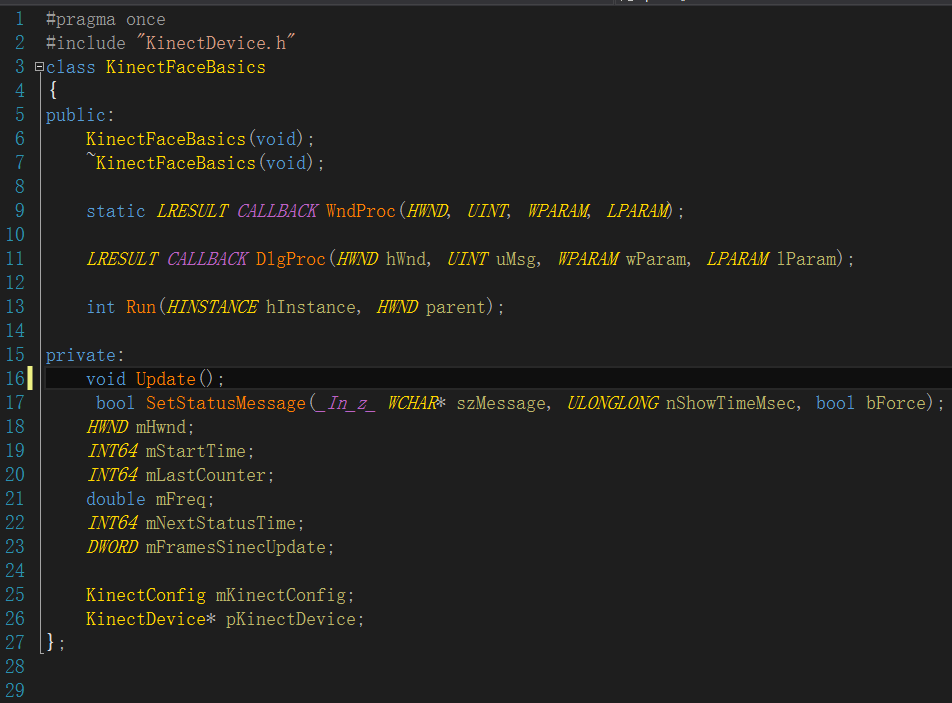


图14

**2. 函数的实现**

在KinectFaceBasics.cpp中，我们来实现声明的函数。首先在文档的开头引入几个头文件：#include “Resource.h”，#include “UITools.h”和#include <strsafe.h>。定义一个宏变量：

#define FACE\_WINDOW\_CLASS\_NAME L"FaceDlgWndClass"

1）初始化参数，在KinectFaceBasics类的构造函数中添加如下的代码：

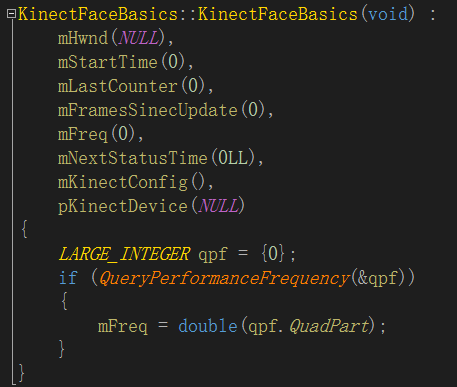


图15

2）Run函数，主要工作是创建窗体并显示，代码如下：

int KinectFaceBasics::Run(HINSTANCE hInstance, HWND parent)

{

EnableWindow(parent, FALSE);

MSG msg = {0};

WNDCLASS wc = UITools::CreateSubWindowClass(FACE\_WINDOW\_CLASS\_NAME, hInstance);

if (!RegisterClass(&wc))

{

EnableWindow(parent, TRUE);

SetForegroundWindow(parent);

return 0;

}

HWND hWnd = CreateDialogParam(

NULL,

MAKEINTRESOURCE(IDD\_FACE),

NULL,

(DLGPROC)KinectFaceBasics::WndProc,

reinterpret\_cast<LPARAM>(this)

);

ScreenSize screenSize = UITools::GetScreenSize();

RECT region;

GetWindowRect(hWnd, &region);

int w = region.right - region.left;

int h = region.bottom - region.top;

int x = (screenSize.width - w) / 2;

int y = (screenSize.height - h) / 2;

MoveWindow(hWnd, x, y, w, h, FALSE);

ShowWindow(hWnd,SW\_SHOW);

while (WM\_QUIT != msg.message)

{

Update();

while (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if (hWnd && IsDialogMessage(hWnd, &msg))

{

continue;

}

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

UnregisterClass(FACE\_WINDOW\_CLASS\_NAME, hInstance);

EnableWindow(parent, TRUE);

SetForegroundWindow(parent);

return 1;

}

3）WndPro函数负责窗体体消息的分发，代码如下：

LRESULT CALLBACK KinectFaceBasics::WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

KinectFaceBasics\* pThis = NULL;

if (WM\_INITDIALOG == uMsg)

{

pThis = reinterpret\_cast<KinectFaceBasics\*> (lParam);

SetWindowLongPtr(hWnd, GWLP\_USERDATA, reinterpret\_cast<LONG\_PTR> (pThis));

}

else

{

pThis = reinterpret\_cast<KinectFaceBasics\*> (GetWindowLongPtr(hWnd, GWLP\_USERDATA));

}

if (pThis)

{

return pThis->DlgProc(hWnd, uMsg, wParam, lParam);

}

return 0;

}

4）DlgProc函数负责窗体消息的处理，代码如下：

LRESULT CALLBACK KinectFaceBasics::DlgProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(wParam);

UNREFERENCED\_PARAMETER(lParam);

switch (uMsg)

{

case WM\_INITDIALOG:

{

}

break;

case WM\_CLOSE:

DestroyWindow(hWnd);

break;

case WM\_DESTROY:

// Quit the main message pump

PostQuitMessage(0);

break;

default:

break;

}

return FALSE;

}

5）SetStatusMessage函数显示状态信息，代码如下：

bool KinectFaceBasics::SetStatusMessage(\_In\_z\_ WCHAR\* szMessage, DWORD nShowTimeMsec, bool bForce)

{

INT64 now = GetTickCount64();

if (mHwnd && (bForce || (mNextStatusTime <= now)))

{

SetDlgItemText(mHwnd, IDC\_KinectFaceStatus, szMessage);

mNextStatusTime = now + nShowTimeMsec;

return true;

}

return false;

}

### 六、识别面部特征

**1. 初始化和配置Kinect设备**

在KinectFaceBasics类的窗体消息处理函数DlgProc中switch代码块中添加kinect设备的初始代码：

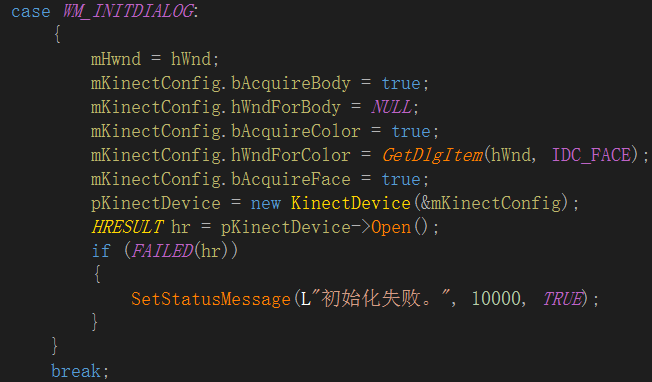


图16

**2. 更新Kinect数据帧**

实现KinectFaceBasics的Update函数，如下所示：

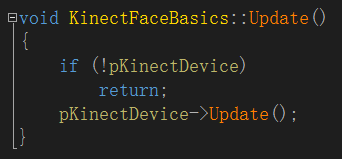


图17

在KinectFaceBasics.cpp的Run函数中添加，对Update函数的调用，如下图所示：

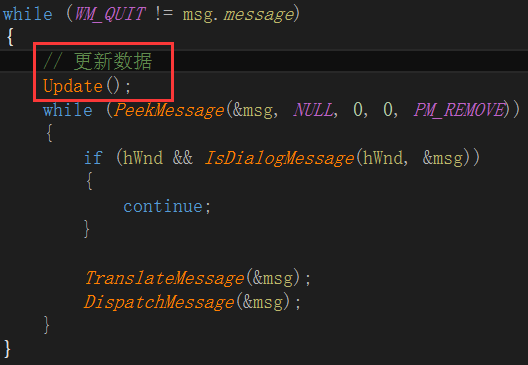


图18

**3. 释放kinect设备**

当我们不需要使用Kinect设备是应该关闭kinect并释放相应资源。在KinectFaceBasics.cpp的析构函数~KinectFaceBasics ()中添加如下代码:

if (pKinectDevice)

{

pKinectDevice->Close();

delete pKinectDevice;

pKinectDevice = NULL;

}

### 七、跳转到Kinect面部特征窗口

1）在KinectFaceFeatures.cpp中引入头文件：#include “KinectFaceBasics.h”。

2）在按钮的点击响应中添加跳转代码，在WndProc函数中添加如下代码：



图19

### 八、编译运行

点击进行编译运行，如果有错误，根据错误提示进行错误的排除。

主界面如下：

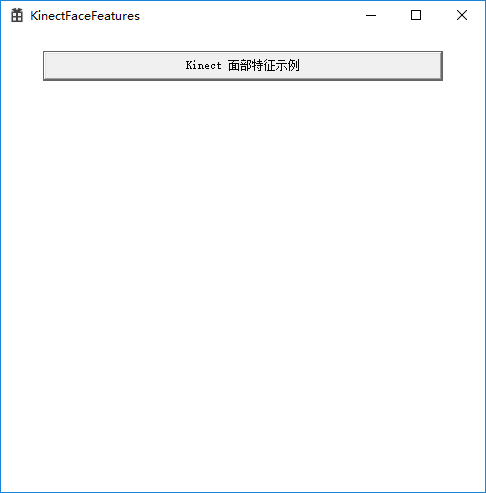


图 20

面部特征界面如下：

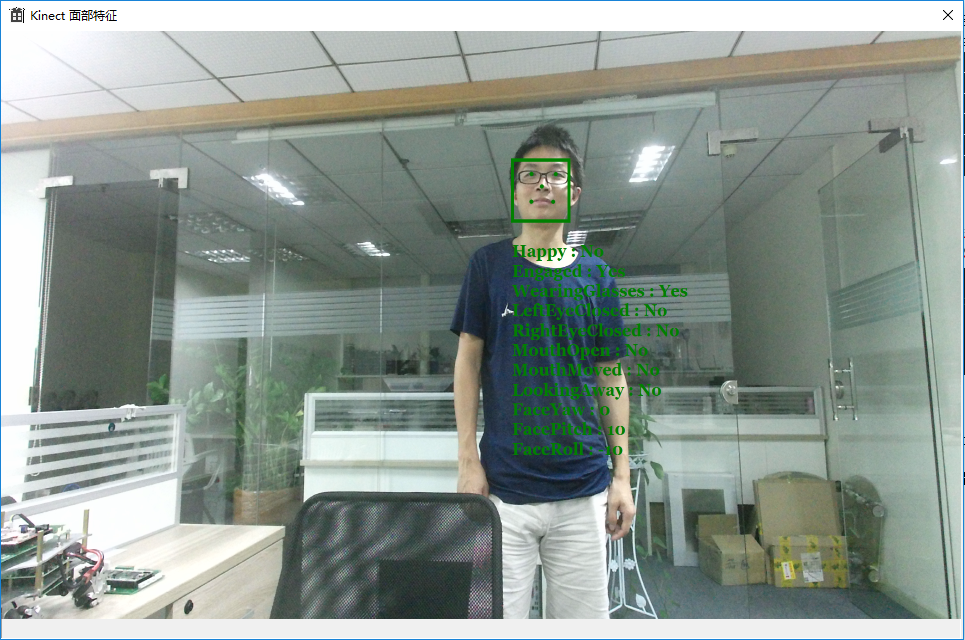


图 21

## 【思考题】

### 选择题

### 2、简答题