```
/*

*/

#include "SceneDrawer.h"
#include <stdio.h>
#include <conio.h>
#include "Kinect.h"
#include <iostream>
#include <windows.h>
#include <winsock2.h>
using namespace std;

static const int BUFSIZE = 5120;
#define ICMD_STOP 0x01
#define ICMD_START 0x02
static int appState = ICMD_STOP;

void sendCommand(int );
void GestureDetector(KinectMsg *);
DWORD WINAPI RemoteServer(void *pArg);
DWORD WINAPI display(void *pArg) ;

DWORD WINAPI display(void *pArg) { //thread for display
 char **argv = (char **) pArg;
 SceneDrawer::glMain(1,argv); //start opengl
 return DWORD(0);
}

/*
detect gesture based upon BodyInfo
One can do his own detection here
*/
void GestureDetector(KinectMsg *pmsg) {
 static BodyInfo pre_body;
 static const int QWAVECOUNT = 4;
 static const float QWAVELEN = 160;
 static int wavecount = 0;
 static float wavesum = 0;
 static bool iLeft = 0;
 BodyInfo cur_body = pmsg->body;
 if(cur_body.bTracking && cur_body.bRHand) {
  //detect start sign
  if(cur_body.pRHand.X - pre_body.pRHand.X > 20)  { //wave right
   if(! iLeft) {//previosue move is go right
    wavesum += cur_body.pRHand.X - pre_body.pRHand.X;
   }
   else {//previouse go left
    //cout<<"total wave left is "<<wavesum<<endl;
    if(wavesum >= QWAVELEN) {
     wavecount ++;
    }
    iLeft = 0;
    wavesum = 0;
   }
  }
  else if(cur_body.pRHand.X - pre_body.pRHand.X < -20){ //wave left
   if(iLeft) {//previosue move is go left
    wavesum += pre_body.pRHand.X - cur_body.pRHand.X;
   }
   else {//previouse go right
    //cout<<"total wave right is "<<wavesum<<endl;
    if(wavesum >= QWAVELEN) {
     wavecount ++;
    }
    iLeft = 1;
    wavesum = 0;
   }
  }
  if(wavecount >= QWAVECOUNT) {//detect continuous hand wave more than QWAVECOUNT times
   wavecount = 0; //reset hand wave count to 0
   sendCommand(ICMD_START); //send start command
  }
 }
 if(cur_body.bTracking && cur_body.bRHand && cur_body.bLHand && cur_body.bRElbow && cur_body.bLElbow) {
  //detect time out sign
  if(abs(cur_body.pRElbow.X - cur_body.pRHand.X) <= 40 &&
   abs(cur_body.pLHand.Y - cur_body.pLElbow.Y) <=40 &&
   CALCDISTANCE(cur_body.pLHand,cur_body.pRHand) <= 50) {
    sendCommand(ICMD_STOP);
  }
  pre_body = cur_body;
 }
}

void sendCommand(int icmd) {
 static int precmd = -1;
 static int count = 0;

 switch(icmd) {
 case ICMD_STOP: //recv a stop cmd
  if(appState != ICMD_STOP) { //prev not stop
   appState = ICMD_STOP;
   cout<<"@@@@@@@@@@@@@@@@@@@@HOME HEALTH APPLICATION STOP@@@@@@@@@@@@@@@@@@@@@@"<<endl;
  }
  break;
 case ICMD_START: //recv a start cmd
  if(appState != ICMD_START) { //prev not start
   appState = ICMD_START;
   cout<<"++++++++++++++++HOME HEALTH APPLICATION START++++++++++++++++++++++"<<endl;
```

```
  }
  break;
 }
 precmd = icmd;
}

DWORD WINAPI RemoteServer(void *pArg) {
 struct hostent *hp;
 struct sockaddr_in saddr, caddr;
 int sock, fromlen, len;
 int port = 19861;
 char recvBuf[100];
 //set up window socket environment
 WSADATA WsaDat;
 if (WSAStartup(MAKEWORD(2,2), &WsaDat) != 0) {
  cerr<<"Couldn't Init"<<endl;
  getch();
  exit(-1);
 }

 if((sock = socket(AF_INET, SOCK_STREAM,0)) <0) {
  cerr<<"sock create failed"<<endl;
  getch();
  exit(-1);
 }

 memset(&caddr, 0, sizeof(caddr));
 memset(&saddr, 0, sizeof(saddr));
 saddr.sin_family = AF_INET;
 saddr.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
 saddr.sin_port = htons(port);

 //bind socket
 if(bind(sock,(LPSOCKADDR)&saddr, sizeof(saddr)) == SOCKET_ERROR) {
  cerr<<"sock bind failed "<<endl;
  perror("Error: ");
  getch();
  closesocket(sock);
  exit(-1);
 }

 //start listening
 if(listen(sock,32)<0) {
  cerr<<"sock listen failed "<<endl;
  perror("Error: ");
  getch();
  closesocket(sock);
  exit(-1);
 }
 int conn = 0;
 while(1) {
  if((conn = accept(sock,NULL,NULL)) >=0) {//get a new connection
   //cout<<"new connection"<<endl;
   while(1) {
    len = recv(conn,recvBuf,BUFSIZE,0);
    if(len == 1) {//recv data length equal to 1
     switch(recvBuf[0]) {
     case ICMD_START:
      sendCommand(ICMD_START);
      break;
     case ICMD_STOP:
      sendCommand(ICMD_STOP);
      break;
     default://receive invalid cmd
     }
    }
    else if(len <= 0) {
     //cout<<"connection down: "<<len<<endl;
     break;
    }
    else {
     recvBuf[len] = '\0';
     //cout<<"recv unknow msg("<<len<<": "<<recvBuf<<endl;
    }
   }
  }
  else {
   cerr<<"socket accept failed"<<conn<<endl;
  }
 }

 closesocket(sock);
 return 0;
}

int main(int argc, char **argv) {

 DWORD glid, rhid;
 //ceate thread for display
 HANDLE _glhandle = CreateThread(0,
  0,
  display, //main body of the thread
  (void*)argv,
  CREATE_SUSPENDED,
  &glid);

 //create thread for host receiving command from HHARemoteClient
 HANDLE _rhhandle = CreateThread(0,
  0,
  RemoteServer, //main body of the thread
```

```
  (void*)argv,
  CREATE_SUSPENDED,
  &rhid);

 Kinect *kinect = new Kinect(1);//create kinect object
 kinect->initKinect(); //init device
 kinect->updateBodyInfo(); //get the first update
 ResumeThread(_glhandle); //start two thead
 ResumeThread(_rhhandle);
 while(1) {
  kinect->updateBodyInfo();
  SceneDrawer::msg = *(kinect->getKinectMsg()); //update image data for display
  GestureDetector(kinect->getKinectMsg()); //do a gesture detection
 }
 return 0;
}
```