|  |
| --- |
| **R E P O R T** |

|  |
| --- |
| **비주얼 프로그래밍**  **기말 기출문제** |



|  |  |
| --- | --- |
| **학 과** | **컴퓨터소프트웨어공학과** |
| **교수님** | **권순각** |
| **학 번** | **20183108** |
| **이 름** | **양윤지** |
| **제출일** | **2019.12.16** |

기출문제 1

서버- 클라이언트 인터넷 연결 상태에서 다른 쪽의 정보를 입력받아 반응하는 프로그램을 구현하시오. 푸시버튼(선그리기)를 추가한다. 한 쪽에서 왼쪽 마우스로 두 개의 서로 다른 위치에 점을 찍고, 선 그리기 버튼을 클릭하면 다른 쪽에 두 점을 잇는 직선이 그려진다.

**변경된 소스**

1) 헤더파일 변수들 추가

CPoint p[2];

int str[1024];

int i = 0;

2) cpp 파일

void Cnetwork2Dlg::OnReceive()

{

int \*pBuf = new int[1025];

int iBufSize = 1024;

int iRcvd;

iRcvd = m\_Csocket.Receive(pBuf, 1024);

if (iRcvd == SOCKET\_ERROR)

{

}

else

{

CClientDC dc(this);

dc.MoveTo(pBuf[0], pBuf[1]);

dc.LineTo(pBuf[2],pBuf[3]);

}

}

void Cnetwork2Dlg::OnLButtonDown(UINT nFlags, CPoint point)

{

// TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

p[i] = point;

i++;

CDialogEx::OnLButtonDown(nFlags, point);

}

void Cnetwork2Dlg::OnBnClickedButton1()

{

// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.

str[0] = p[0].x;

str[1] = p[0].y;

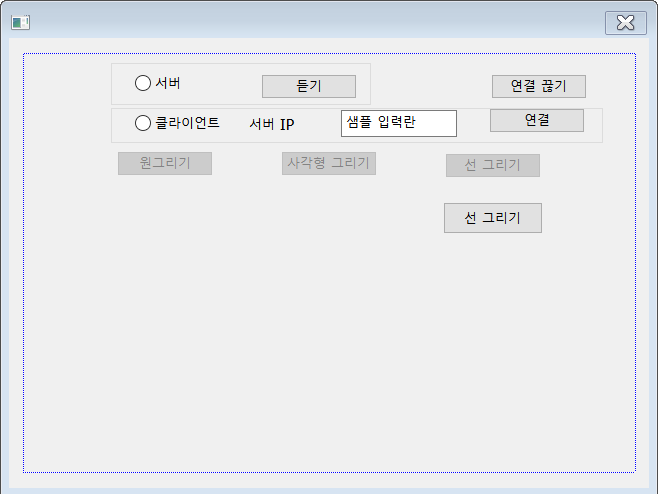
str[2] = p[1].x;

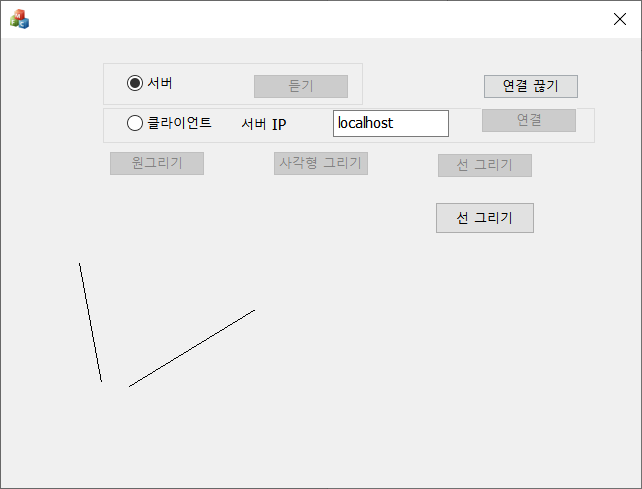
str[3] = p[1].y;

i = 0;

m\_Csocket.Send(str, 1024);

}

**모달 창**

**실습 결과**

2. 대화상자 기반으로 프로젝트(EXDLL)를 생성한다. 대화상자에는 푸시버튼(환영)을 배치한다. 환영 버튼을 클릭하면, “홍길동(본인이름)입니다. DLL 구현하였습니다.” 메시지 박스를 출력시킨다. 메시지 박스를 출력하는 것은 DLL로 구현한다. DLL 프로젝트의 이름은 MSGDLL로 정한다.

**변경된 소스**

// EXDLLDlg.cpp: 구현 파일

//

#include "stdafx.h"

#include "EXDLL.h"

#include "EXDLLDlg.h"

#include "afxdialogex.h"

#include "MSGDLL.h" // 추가된 헤더 dll

void CEXDLLDlg::OnBnClickedButton1()

{

// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.

CString str = PowM();

AfxMessageBox(str);

}

-----------------------------------------------------------------------------------------------

// MSGDLL.h : MSGDLL DLL의 주 헤더 파일

//

extern "C" \_declspec (dllimport) CString PowM();

// MSGDLL.cpp : DLL의 초기화 루틴을 정의합니다.

//

#include "stdafx.h"

#include "MSGDLL.h"

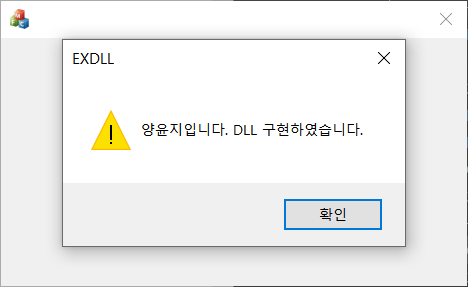
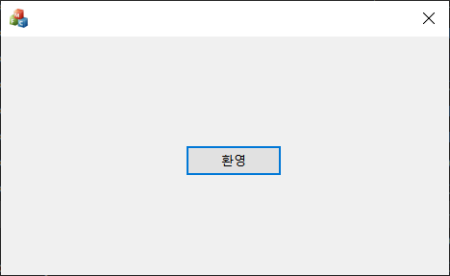
extern "C" \_declspec (dllexport)CString PowM() {

CString str;

str.Format(\_T("양윤지입니다. DLL 구현하였습니다."));

return str;

}



3. 대화상자 기반으로 프로젝트를 생성한다. 대화상자에는 푸시버튼1(시작), 푸시버튼2(종료) 두개를 배치한다. 체크박스(잠시 멈춤)도 배치한다. 시작 버튼을 클릭하면, 첫 번째 선이 시작점(수직좌표는 50, 수평좌표는 50)에서 끝점(수직좌표는 50, 수평좌표는 250)까지 그려진다. 선은 0.1초마다 수직방향으로 50 이동하여 같은 길이의 선이 그려진다. 즉, 두 번째 선은 시작점(수직좌표는 100, 수평좌표는 50)에서 끝점(수직좌표는 100, 수평좌표는 250)까지 그려진다. 선이 10개까지 그려지고, 10개가 되면 0.1초 마다 하나씩 감소한다. 선이 없어지면 0.1초 지나서 다시 첫 번째 선이 그려진다. 이러한 작업은 계속적으로 반복된다. 푸시버튼 ‘종료’를 클릭하면 쓰레드가 종료되고, 프로그램이 종료된다. 또한, 체크박스 “잠시 멈춤” 클릭하면 쓰레드가 잠시 정지해 있고, 다시 체크박스를 클릭하면 쓰레드가 다시 돌아간다.

// exam03Dlg.h: 헤더 파일

//

public:

afx\_msg void OnBnClickedButton2();

static UINT ThreadWorker1(LPVOID pParam);

CWinThread \*m\_pWorkerThread1 =NULL;

int line = 0 ;

int gap = 1;

BOOL incr = FALSE;

afx\_msg void OnBnClickedButton1();

afx\_msg void OnBnClickedCheck1();

BOOL m\_bWorkerPause1;

// exam03Dlg.cpp: 구현 파일

//

#include "stdafx.h"

#include "exam03.h"

#include "exam03Dlg.h"

#include "afxdialogex.h"

void Cexam03Dlg::OnPaint()

{

else

{

CDialogEx::OnPaint();

CClientDC dc(this);

if (incr == TRUE) {

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

line = 50 \* i;

dc.MoveTo(50, line);

dc.LineTo(250, line);

}

}

}

}

UINT Cexam03Dlg::ThreadWorker1(LPVOID pParam) {

// 위쪽 오른쪽

Cexam03Dlg \*pWnd = (Cexam03Dlg\*)pParam;

BOOL tf = true;

while (1) {

pWnd->incr = TRUE;

if (tf) {

pWnd->Invalidate(TRUE);

pWnd->gap++;

Sleep(100);

if (pWnd->gap == 10) {

tf = false;

}

}

else {

pWnd->gap--;

pWnd->Invalidate(TRUE);

Sleep(100);

if (pWnd->gap == 0) {

tf = true;

}

}

}

return 0;

}

void Cexam03Dlg::OnBnClickedButton2()

{

// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.

if (!m\_pWorkerThread1)

{

m\_pWorkerThread1 = AfxBeginThread(ThreadWorker1, this);

}

}

void Cexam03Dlg::OnBnClickedButton1()

{

// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.

if (m\_pWorkerThread1) {

HANDLE hThread1 = m\_pWorkerThread1->m\_hThread;

WaitForSingleObject(hThread1, 0);

m\_pWorkerThread1 = NULL; }

OnOK();

}

void Cexam03Dlg::OnBnClickedCheck1()

{

// TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.

if (m\_bWorkerPause1 == FALSE) {

m\_bWorkerPause1 = TRUE;

m\_pWorkerThread1->SuspendThread();

}

else {

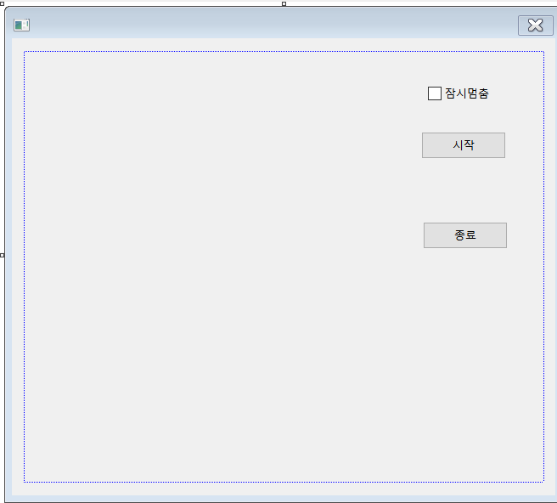
m\_bWorkerPause1 = FALSE;

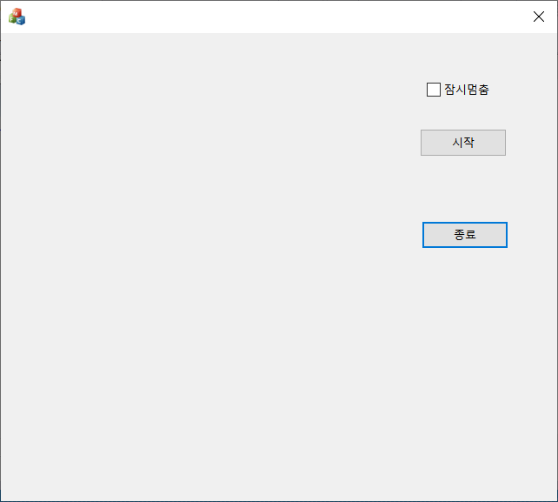
m\_pWorkerThread1->ResumeThread();

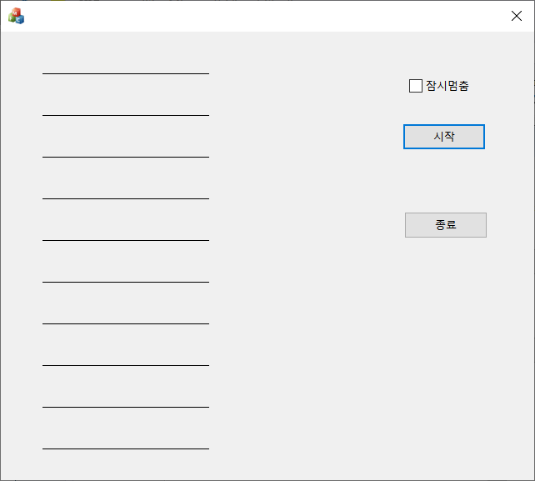
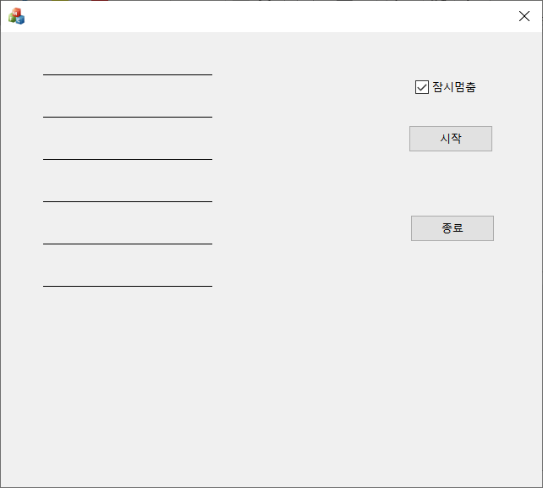
}

}

**컨트롤 배치**







4. 단일 문서 기반으로 프로젝트를 생성한다. 윈도우 화면에 마우스 왼쪽 버튼을 누르면 해당되는 위치에 반지름 10인 원이 그려진다. 또한,계속해서 다른 위치에 왼쪽 버튼을 누를 때마다 원이 그려진다. 그리고, 작업한 내용이 저장가능하게 구현하며, 20개 이상 원을 그려서 작업한 내용을 프로젝트 폴더안에 붙인 이름의 파일명으로 저장해놓는다.

// CDots.h

#pragma once

#include <afx.h>

class CDots :

public CObject

{

DECLARE\_SERIAL(CDots)

public:

CDots();

~CDots();

CDots(CPoint ptF);

void Draw(CDC \*pDC);

void Serialize(CArchive &ar);

private:

CPoint m\_pt;

};

// CDots.cpp

#include "stdafx.h"

#include "CDots.h"

IMPLEMENT\_SERIAL(CDots,CObject,1) // 세미콜론 하면 안됨

//CDots를 직렬화하겠다.

CDots::CDots()

{

}

CDots::~CDots()

{

}

CDots::CDots(CPoint ptF) {

m\_pt = ptF;

}

void CDots::Draw(CDC \*pDC) {

pDC->Ellipse(m\_pt.x - 10, m\_pt.y - 10, m\_pt.x + 10, m\_pt.y + 10);

}

void CDots::Serialize(CArchive &ar) {

if (ar.IsStoring()) {

ar << m\_pt;

}

else {

ar >> m\_pt;

}

}

// YYJSDDIDoc.h: CYYJSDDIDoc 클래스의 인터페이스

//

#include "CDots.h"

public:

virtual BOOL OnNewDocument();

virtual void Serialize(CArchive& ar);

private:

CObArray m\_oaDots; // 만능배열

public:

CDots\* AddDots(CPoint ptF);

CDots\* GetDots(int nIndex);

int GetDotsCount();

// YYJSDDIDoc.cpp: CYYJSDDIDoc 클래스의 구현

//

BOOL CYYJSDDIDoc::OnNewDocument()

{

if (!CDocument::OnNewDocument())

return FALSE;

// TODO: 여기에 재초기화 코드를 추가합니다.

// SDI 문서는 이 문서를 다시 사용합니다.

int liCount = m\_oaDots.GetSize();

if (liCount) {

for (int i = 0; i < liCount; i++)

delete m\_oaDots[i];

m\_oaDots.RemoveAll();

}

return TRUE;

}

// CYYJSDDIDoc serialization

void CYYJSDDIDoc::Serialize(CArchive& ar)

{

m\_oaDots.Serialize(ar); // 열기하면 여기로 항상 옴

}

CDots\* CYYJSDDIDoc:: AddDots(CPoint ptF) {

CDots\* pDots = new CDots(ptF);

m\_oaDots.Add(pDots);

SetModifiedFlag(); // view의 OnDraw를 호출한다.

return pDots; // 현재 포인터 값을 반환한다.

}

CDots\*CYYJSDDIDoc:: GetDots(int nIndex) {

return (CDots\*)m\_oaDots[nIndex];

}

int CYYJSDDIDoc::GetDotsCount() {

return m\_oaDots.GetSize();

}

// YYJSDDIView.h: CYYJSDDIView 클래스의 인터페이스

//

private:

CPoint ptF;

BOOL isNewDots;

// YYJSDDIView.cpp: CYYJSDDIView 클래스의 구현

//

#include "YYJSDDIDoc.h"

#include "YYJSDDIView.h"

void CYYJSDDIView::OnDraw(CDC\* pDC) // view가 doc클래스를 보기위한 작업

{

CYYJSDDIDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 여기에 원시 데이터에 대한 그리기 코드를 추가합니다.

int liCount = pDoc->GetDotsCount(); // 점의 쌍점이 몇개 있는가

if (liCount) {

CDots \*ptDots;

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

ptDots = pDoc->GetDots(i);

ptDots->Draw(pDC);

}

}

isNewDots = TRUE;

}

void CYYJSDDIView::OnLButtonUp(UINT nFlags, CPoint point)

{

// TODO: 여기에 메시지 처리기 코드를 추가 및/또는 기본값을 호출합니다.

ptF = point;

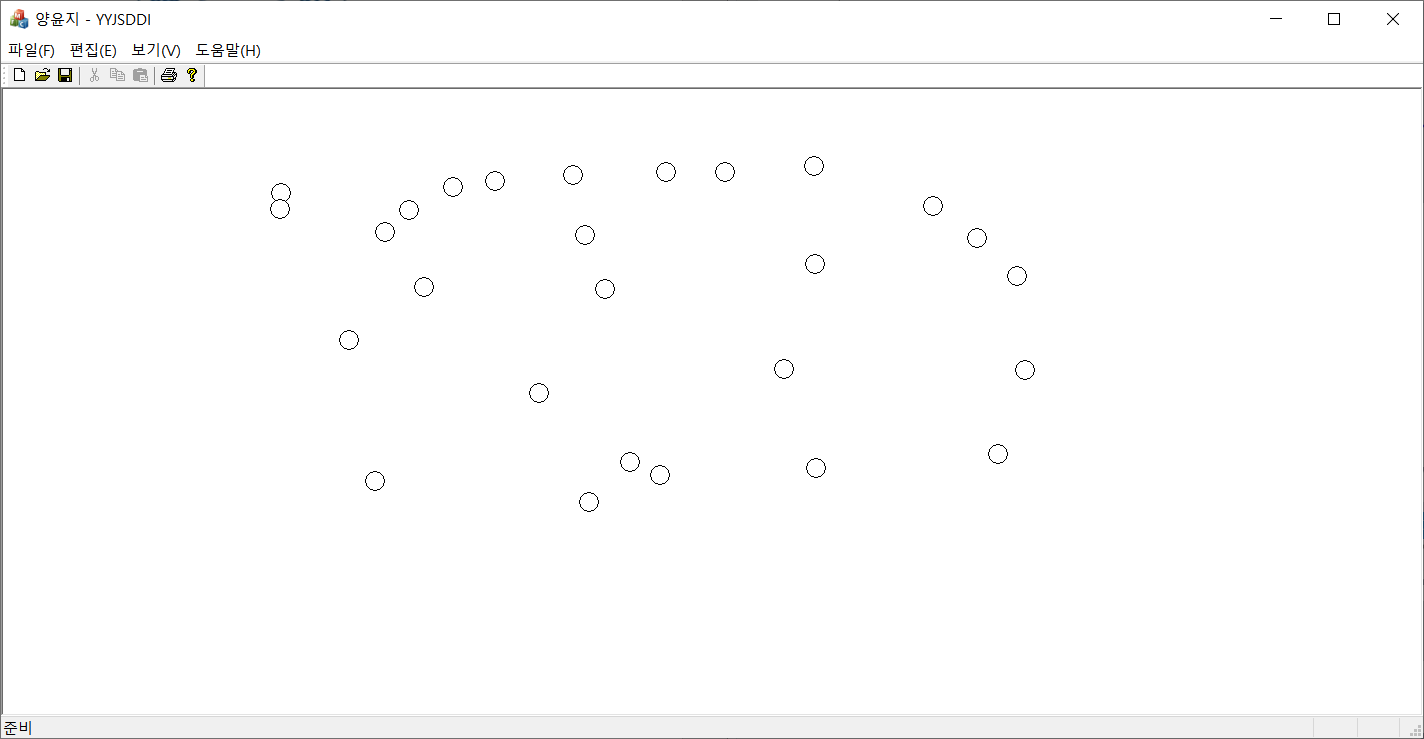
CClientDC dc(this);

CDots \*pDots = GetDocument()->AddDots(ptF); // view가 doc보려고 하는 작업

pDots->Draw(&dc);

CView::OnLButtonUp(nFlags, point);

}



5. 팀프로젝트에서 사용한 클래스 5개를 나열하고, 각 클래스의 용도에 대하여 설명하시오.

|  |  |
| --- | --- |
| 클래스 | 기능 |
| CExplain  (게임 설명) | * 게임에 대한 전면적인 설명을 보여준다. |
| CmfcteamprojectDlg  (게임 시작화면) | - OnPaint() : 바탕에 깔린 비트맵을 보여준다.  - OnLButtonDown(UINT nFlags, CPoint point) : 게임 설명과 게임 시작, 랭킹을 클릭 시 볼 수 있도록 한다.   * ONCtlColor(CDC\* pDC, CWnd\* pWnd, UINT nCtlColor) : static text의 배경색을 투명하게 바꾼다. |
| CMysocket  (소켓 연결) | - OnAccept(int nErrorCode) : 클라이언트의 Connect()함수를 호출 시, 서버에서 호출되는 함수   * OnClose(int nErrorCode) : 상대방이 Close() 함수를 호출한 경우 호출되는 함수 * OnConnect(int nErrorCode) : 클라이언트에서 Connect()함수를 호출 했을 때, 이 시도에 대한 결과(성공적 연결/연결 실패)가 나왔을 때 호출되는 함수 * OnReceive(int nErrorCode) : 상대방이 Send() 함수를 호출하여 전송한 데이터가 도착한 경우 호출되는 함수. 이 때, 이 함수 내에서 Receive() 함수는 단 한번만 호출함 * OnSend(int nErrorCode) : 클라이언트에서 OnConnect(), Connect()함수가 호출된 이후, 서버측에서 OnAccept(), Accept()함수가 호출된 이후 호출 되는 함수 |
| CPOPUP (솔로와 듀얼모드 선택) | * ONInitDialog() : 다이얼로그에 보이는 static text의 배경을   투명화시킨다.   * OnPaint() : 바탕에 깔린 비트맵을 보여준다. * OnCtlColor() : static text의 배경색을 투명하게 바꾼다. * OnBnClickedOK() : 솔로 라디오버튼을 클릭시 솔로 모드 게임으로 연결, 듀얼 라디오버튼을 클릭시 네트워크 연결 화면으로 간다. |
| CRANK  (개인의 점수 랭킹) | * Load() : rank.txt을 파일 출력을 통해 현재 모든 랭킹을 보여준다. |
| NETWORK  (듀얼모드 연결) | * OnInitDialog() : 다이얼로그에 보이는 static text, Edit Box의 배경을 투명화시킨다. * OnRType() : 서버와 클라이언트 라디오버튼 클릭시 버튼 비/활성화 * OnAccept() : 서버에서 호출되는 함수 * OnConnect() : 클라이언트에서 연결 호출 시도 시 연결 혹은 실패 * OnClose() : 연결 취소 * OnBnClickedBlistendang() : 서버의 듣기 버튼 클릭시 클라이언트와 연결 * OnBnClickedBconnect() : 클라이언트의 연결 클릭 시 서버와 연결 * OnBnClickedButton1() : 확인 버튼 클릭 시 듀얼 모드 게임으로 연결 |

|  |  |
| --- | --- |
| 클래스 | 기능 |
| CardDialog  (보물상자 클릭 시 랜덤 카드) | * OnInitDialog() : 카드 비트맵의 초기 위치를 정하고, 랜덤 값을 얻을 수 있다. * OnPaint() : 카드 비트맵들을 화면에 보여준다. * OnLButtonDown(UINT nFlags, CPoint point) : 5개의 랜덤한 카드(꽝, 속도 증가, 시간 추가)를 랜덤하게 섞어서 카드를 선택한 후 그 값을 게임에 반영한다. |
| CPopup  (기록 등록) | * 게임 종료 시 개인의 기록을 기록할 수 있다. |
| CPowDlg2  (싱글 게임의 dll이다.) | * CPowDlg2(CWnd \*pParent = nullptr) : 캐릭터의 초기 값을 정한다. * bigisland() : 노랑 장애물(공)에 캐릭터가 맞았다면 섬의 크기가 커져 사용자가 피할 공간이 커진다. * eraser(CRgn & rgn, BOOL tf) : 장애물이 사라지는 함수 * EXIT() : 목숨이 없다면 장애물들을 모두 지운다. * first\_gap(int num, int I, int x, int m) : 장애물의 처음 값을 정한다. * life() : 캐릭터가 장애물에 맞았을 때 목숨이 감소한다. * -OnInitDialog() : 장애물(쓰레드)들과 보물상자, 갈매기, 생명의 위치를 정한다. * ONnKeyUp(UNIT nChar,UNITCnt,UINT nFlags) : 캐릭터가 A,W,S,D를 통해 움직일 수 있다. 또한 섬의 경계를 벗어날 수 없다. * OnLButtonDown(UNIT nFlags,CPoint point) : 갈매기와 보물상자가 나타났을 때 클릭할 수 있다. * OnPaint() : 섬, 바다, 생명, 캐릭터, 갈매기 등 비트맵을 화면에 보여준다. * OnTimer() : 갈매기, 보물상자, 쓰레드들을 시간에 맞게 호출한다. * Paint(int num, int location, int select) : 쓰레드에서 호출한 장애물(원)을 select(랜덤 색)에 따라 그린다. * random() : 장애물(원)의 위치를 랜덤으로 지정한다. * smallisland() : 검정 장애물(공)에 캐릭터가 맞았다면 섬의 크기가 작아져서 사용자가 피할 공간이 작아진다. * turn() : 노란 공과 검정 공에 맞은 후 일정 시간 이후 다시 원래 경계 값으로 돌려준다. * score() : 장애물에 맞으면 점수가 감소된다. 무적상태 시 점수는 감소 되지 않는다. * show() : 카드선택시 선택한 값을 게임에 반영한다. |

|  |  |
| --- | --- |
| 클래스 | 기능 |
| CPowDlg  (듀얼 게임의 dll이다.) | * CPowDlg(CWnd \*pParent = nullptr) : 캐릭터의 초기 값을 정한다. * bigisland() : 노랑 장애물(공)에 캐릭터가 맞았다면 섬의 크기가 커져 사용자가 피할 공간이 커진다. * eraser(CRgn & rgn, BOOL tf) : 장애물이 사라지는 함수 * EXIT() : 목숨이 없다면 장애물들을 모두 지운다. * first\_gap(int num, int I, int x, int m) : 장애물의 처음 값을 정한다. * life() : 캐릭터가 장애물에 맞았을 때 목숨이 감소한다. * -OnInitDialog() : 장애물(쓰레드)들과 보물상자, 갈매기, 생명의 위치를 정한다. * ONnKeyUp(UNIT nChar,UNITCnt,UINT nFlags) : 캐릭터가 A,W,S,D를 통해 움직일 수 있다. 또한 섬의 경계를 벗어날 수 없다. * OnLButtonDown(UNIT nFlags,CPoint point) : 갈매기와 보물상자가 나타났을 때 클릭할 수 있다. * OnPaint() : 섬, 바다, 생명, 캐릭터, 갈매기 등 비트맵을 화면에 보여준다. * OnTimer() : 갈매기, 보물상자, 쓰레드들을 시간에 맞게 호출한다. * Paint(int num, int location, int select) : 쓰레드에서 호출한 장애물(원)을 select(랜덤 색)에 따라 그린다. * random() : 장애물(원)의 위치를 랜덤으로 지정한다. * smallisland() : 검정 장애물(공)에 캐릭터가 맞았다면 섬의 크기가 작아져서 사용자가 피할 공간이 작아진다. * turn() : 노란 공과 검정 공에 맞은 후 일정 시간 이후 다시 원래 경계 값으로 돌려준다. |
| DualDialog  (보물상자 클릭 시 랜덤 카드) | * OnInitDialog() : 카드 비트맵의 초기 위치를 정하고, 랜덤 값을 얻을 수 있다. * OnPaint() : 카드 비트맵들을 화면에 보여준다. * OnLButtonDown(UINT nFlags, CPoint point) : 5개의 랜덤한 카드(꽝, 속도 증가, 시간추가, 상대방 화면 가리기, 상대방 시간 증가)중에서 3개의 랜덤한 카드를 통해 카드를 선택한 후 그 값을 게임에 반영한다. |