**비주얼프로그래밍**

**요소설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Visual Programming(같은 그림 찾기 게임)* |
| 팀 명 | *09* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 2013-11-05 |

|  |  |
| --- | --- |
| **팀원** | 김 성근 (조장) |
| 김 현준 |
| 이 종석 |
| **지도교수** | 김 상철 교수 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 비주얼프로그램밍 수강 학생 중 프로젝트 “같은 그림 찾기 게임”를 수행하는 팀 “09”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “김 성근, 김 현준, 이 종석”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 중간보고서-같은그림찾기게임.doc |
| **원안작성자** | 김 성근 |
| **수정작업자** | 김 현준, 이 종석 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **수정날짜** | **대표수정자** | **Revision** | **추가/수정 항목** | **내 용** |
| 2013-11-05 | 김성근 | 1.0 | 최초 작성 |  |
| 2013-11-12 | 김현준 | 1.1 | 내용 수정 | 수행 내용 추가 |
| 2013-11-19 | 이종석 | 1.2 | 내용 수정 | 수행 내용 추가 |
| 2013-11-23 | 김성근 | 2.0 | 최종 작업 | 최종보고서 작성 및 검토 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[**1** **프로젝트 목표** 4](#_Toc230855220)

[**2** **수행 내용 및 중간결과** 5](#_Toc230855221)

[**2.1** **계획서 상의 연구내용** 5](#_Toc230855222)

[**2.2** **수행내용** 5](#_Toc230855223)

[**3** **최종보고서 본문** 7](#_Toc230855224)

[**4** **자기평가** 8](#_Toc230855225)

# **프로젝트 목표**

Visual Programming 프로젝트에서는 마이크로소프트에서 제공해주는 MFC(Microsoft Foundation Class)를 사용하여 실제로 사용 가능한 MFC 응용프로그램을 제작한다. 이 응용프로그램은 같은 그림 찾기 게임을 기반으로 해서 만들어진다. 이 게임은 Visual C++ MFC 윈도우 응용프로그램(성윤정 지음)에서 제공되는 코드이며, 이 코드를 이용하여 한 학기 동안 배운 다양한 MFC API를 이용하여 해당 프로그램을 확장한다.

응용프로그램을 만들어 가는 과정을 통하여 MFC에서 발생하는 메시지인 Window Message, Command Message, Control Notification Message에 대해서 이해하고 처리할 수 있는 능력을 향상시키고, MFC 응용 프로그램의 기본 골격을 이루는데 절대적인 클래스 4가지의 CWinApp, CDocument, CFrameWnd, CView 관계를 이해하도록 한다. 이런 과정을 통하여 추후 Ios Application이나 Android Application 개발에 필요한 개념을 이해하고 적용할 수 있는 역량을 키우도록 한다.

# **수행 내용 및 중간결과**

## **계획서 상의 연구내용**

1. 게임모드를 연습모드와 랭킹모드 두 가지를 할 수 있도록 구현한다.  
   - 메뉴 구성  
     
     
   - 랭킹 모드  
    랭킹 모드는 가장 처음 2 X 3 박스로 게임을 시작하여 5 X 6 박스로 게임이 끝나는 시간을 측정해서 기록하는 모드이다. 따라서 랭킹 모드를 구현하기 위해서는 WM\_TIMER를 이용한 스톱워치가 기본적으로 구현이 되어 있어야 한다. 그리고 랭킹을 계속 기록하기 위한 공간도 필요하다.  
     
   - 연습 모드  
    연습 모드는 고급( 5 X 6 ), 중급( 4 X 5 ), 초급( 3 X 4 )으로 나뉘어 지며, 게임이 시작되어 끝날 때 까지 시간을 측정하여 기록하는 모드이다. 랭킹 모드와 동일하게 기록 측정을 위하여 스톱워치가 필요하며 난이도 조절을 위한 과정도 필요하다.

게임 - 새 게임

- 게임모드 - 랭킹모드

- 연습모드 - 고급

- 중급

- 초급

1. 사용자가 선택할 수 있는 모양을 제공한다.   
   - 메뉴 구성  
     
     
   - 구현  
    각 메뉴별로 이미지를 최대 6 X 6 (36)개를 준비해야 한다. 또한 게임 처음 시작 시에 안보이게 할 빈 이미지도 한 개 준비해야 한다. BITMAP 입력 시에 각 메뉴별 이미지를 순차적으로 입력하여 하나의 1부터 36까지의 숫자 그룹을 형성할 수 있도록 한다.

게임 - 모양변경 - 둘리  
 - 포켓몬스터  
 - 원피스

1. 통계기능을 제공한다.  
   - 메뉴 구성  
     
     
   - 구현  
    통계는 Modal 형태인 Dialog로 랭킹모드와 연습모드를 따로 따로 볼 수 있도록 구현한다. 또한 연습모드에서도 초급, 중급, 고급을 ListBox를 통하여 선택적으로 볼 수 있게 한다. 가장 빠른 기록 순서대로 나열되어 보여지며, 1등부터 5등까지만 보여지도록 구현 한다. 기록되는 것은 시간과 사용자 명이다. ( 사용자 명은 4번 구현인 Welcome Dialog를 통하여 쓰여진다. ) 구현 방법은 CFile을 이용하거나 Sealize 함수를 이용하여 파일 입출력을 통해 HardDisk에 기록 될 수 있도록 한다.

게임 - 통계

1. 게임 처음 시작 시 실행할 게임모드와, 사용자명을 입력받은 Dialog를 띄운다.  
   - 구현  
    사용자 명은 초기값으로 Window 계정 이름을 가져오도록 한다. 만약 빈칸으로 입력한다면 Guest가 사용자 명에 입력되도록 한다. 게임모드는 앞에서 설정한 것과 동일하게 랭킹모드, 연습모드를 선택 가능하게 하며, 연습모드일 경우에는 난이도 선택이 가능 하도록 구현한다.
2. 네트워크를 이용하여 대전할 수 있는 기능을 추가한다.  
   - 메뉴 구성  
     
     
   - 구현  
    사용자가 대전모드를 클릭하면 Modal 형식인 Dialog가 보여지고, 다른 사용자가 같은 상황에서 IP를 통하여 대전을 신청할 수 있도록 구현한다. 대전 신청이 된 상태에서 대전 신청을 요청한 상대가 게임 시작을 하여 사용자간 게임이 할 수 있도록 한다. 게임 진행은 같은 그림을 찾게 되면 계속해서 자신 순서를 유지하고 틀린 경우 상대편에게 순서가 넘어가는 것을 반복하여 모두 그림을 찾을 때까지 유지하도록 구현한다. 게임이 진행되는 과정에서만 채팅 메뉴를 클릭하여 상대방과 대화를 할 수 있도록 구현한다.

게임 - 새 게임

- 게임모드 - 랭킹모드

- 연습모드

- 대전모드(추가)

대전 - 채팅

- 기권

## **수행내용**

1. 난이도 고급, 중급, 초급을 선택 할 수 있도록 구현한다.   
   (1). 리소스 뷰를 이용해서 IDR\_MAINFRAME의 메뉴를 계획한 내용 **1.**과 같이 수정.  
   (2). 고급, 중급, 초급 모드에 대한 ID 추가   
    ( ID\_GRADE\_TOP , ID\_GRADE\_MIDDLE, ID\_GRADE\_LOW )  
   (3) 각각에 대한 COMMAND와, UPDATE\_COMMAND\_UI 이벤트 처리기를 CGameView에 추가 및 구현  
   (4) CGameDoc에 난이도를 표현할 enum을 추가.  
     
   (5) 난이도 변수를 저장할 int m\_nGrade를 Doc에 추가하고, 초기값을 TOP으로 지정. View에서 접근 가능하도록 Get, Set 함수 구현  
   (6) 각각 이벤트 처리기가 수행 될 때 게임이 재 시작이 될 수 있도록 구현. Doc 클래스에 구현.

enum enGameGrade{

TOP,

MIDDLE,

LOW

};

1. 랭킹모드와 연습모드를 구분 할 수 있도록 구현한다.  
   (1) **수행내용 1.** 에서 추가한 메뉴를 바탕으로 메뉴 구성  
   (2) 랭킹모드에 ID 부여 및 이벤트 처리기 추가. ( ID\_MODE\_RANKING )  
   (3) 게임모드 구분을 위해 CGameDoc에 enum을 추가.  
     
   (4) 게임모드 변수를 저장할 int m\_nMode를 Doc에 추가하고 초기값을 PRACTICE로 지정. View 에서 접근 가능하도록 Get, Set함수 구현  
   (5) 각 이벤트 처리기가 수행되도록 구현   
   - 게임모드 선택 함수의 일반적인 구현

enum enGameMode{

PRACTICE,

RANKING

};

void CGameView::OnUpdateGradeMiddel(CCmdUI \*pCmdUI){

CGameDoc \*pDoc = GetDocument();

if(pDoc->GetMode() == PRACTICE)

pCmdUI->SetCheck(pDoc->GetGrade() == MIDDLE);

}

void CGameView::OnGradeMiddel(){

CGameDoc \*pDoc = GetDocument();

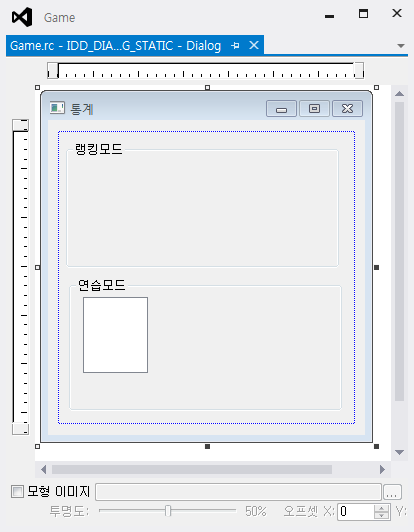
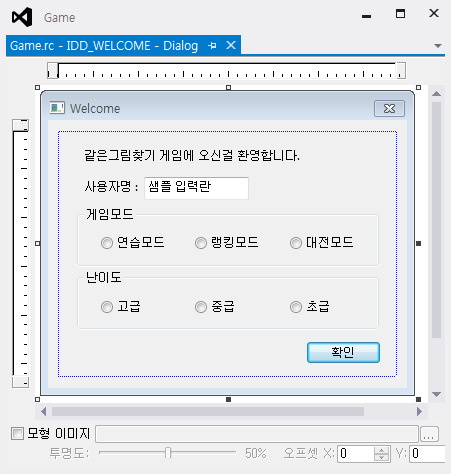
pDoc->SetMode(PRACTICE);

pDoc->SetGrade(MIDDLE);

pDoc->UpdateGrade();

Invalidate();

}

1. 기록 저장을 위하여 스톱워치 구현  
   (1) 스톱워치는 파일로 저장되어야 함으로 Doc 함수에 int형 변수 m\_nH, m\_nM, m\_nS, m\_nTimest 4가지를 선언하고 Getter, Setter 함수를 작성한다.  
   (2) View Class에서 StopWatch 클래스를 선언하고 해당함수를 구현한다.   
   (3) WM\_TIMER 가 발생되게 하기 위해서 게임시작할 때 SetTimer()가 실행될 수 있도록 구현 및 게임 종료가 되면 KillTimer()를 하도록 구현.
2. 기록 저장구현.  
   (1) CGameDoc 클래스에 OnWriteScoreFile( FileName, Mode ) 함수 구현. ( mode와 fileName을 파리미터로 주는 이유는 랭킹모드와 연습모드에 대해서 각각 파일 저장을 따로 하기 위함)  
   (2) String 저장을 하기 위해 CStdioFile을 사용함.   
   (3) 사용자 이름과 기록을 ‘ , ‘ 로 구분하여 작성.
3. 통계 구현  
   (1) 계획한 내용 **3.**과 같이 메뉴 구성에 통계 추가  
   (2) ID\_STATICS\_RANKING 라고 ID를 주고 이벤트 처리기 추가.   
   (3) 통계를 띄워서 보여줄 DIALOG 구현  
     
   (4) listBox에 control 변수 m\_ctrPList 추가, 기록이 출력될 static text 공간을 각각 m\_strRankList, m\_strPractice 추가  
   (5) 파일일 읽어오기 위해서 OnReadScoreFile( FileName, mode) 추가   
   (6) 파일로부터 데이터를 읽어와서 기록을 밀리 초 단위로 계산 하여 정렬. 정렬하기 위해 Record라는 클래스를 생성해 시간과 사용자 이름을 저장하고, 시간을 밀리 초 단위로 계산하는 함수가 있도록 구성하고, Array<Record, Record>를 이용하여 정렬  
   (7) m\_ctrPList에 고급, 중급, 초급이 있도록 InitDialog시 추가해주고, 더블 클릭하면 함수가 실행되어 기록을 출력하도록 변경.
4. 처음 게임 시작화면 Dialog 구현  
     
   (1) 게임모드와 난이도를 선택하고 사용자 명을 입력 할 수 있도록 Dialog 구성.  
   (2) 랭킹모드나 대전모드를 클릭할 시에 난이도가 선택되지 않도록 비활성화 시킴.  
   (3) 사용자명에 Default로 Window 계정명을 가져오도록 처리  
     
   (4) 가장 먼저 띄우도록 Create 문에서 선언 후 Modal()형식으로 출력

DWORD dwCnt = 0;

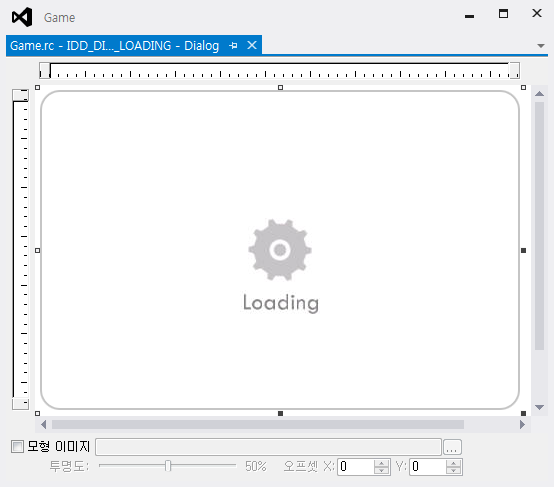
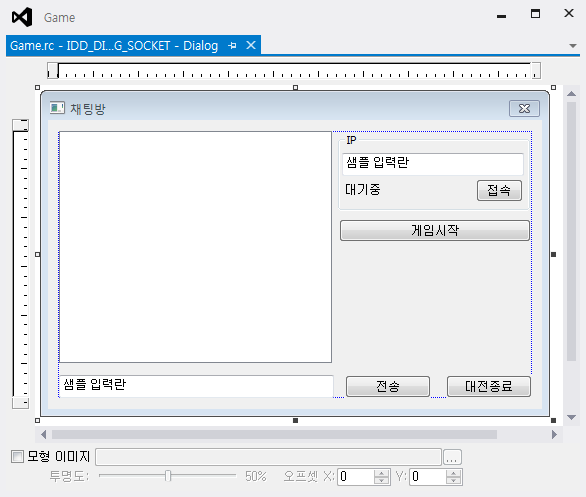
LPTSTR pStr = NULL;

WTSQuerySessionInformation(

WTS\_CURRENT\_SERVER\_HANDLE, WTS\_CURRENT\_SESSION,

WTSUserName, &pStr, &dwCnt);

m\_strName = pStr;

1. 가장 처음 윈도우가 생성되기 전에 2초동안 로딩화면 띄움( 추가 )  
     
   (1) 위와 같은 Dialog를 Bitmap을 추가하여 IDD\_DIALOG\_LOADING 이름으로 Dialog 생성.  
   (2) 프로그램 시작 전 2초간 SetTimer를 이용하여 Dialog를 실행시킨다.
2. 사용자 선택 모양 제공  
   (1) BITMAP 이미지 생성   
   (2) 각각 파트에 맞게 BITMAP 이미지 리소스에 추가. 각 파트의 가장 처음 BITMAP은 그림을 가리기 위한 BITMAP이다.  
   (3) 계획내용 2와 같이 메뉴 구성 추가 및 이벤트 처리기 추가 ( ID\_TYPE\_DOOLY , ID\_TYPE\_POCKETMON, ID\_TYPE\_ONEPIECE)   
   (4) 모양이 변경된다면 게임이 재 시작 하도록 구현.
3. 네트워크 대전 기능 추가를 위한 소켓 구성  
   (1) 소켓을 사용하기 위하여 header 파일 sdafx.h에 **#include <afxsock.h>** 추가  
   (2) CSocket을 상속받는 클래스 CServerSocket과 CDataSocket을 작성한다. CServer는 서버역할을 하기 위한 클래스이고 CData는 Data를 주고 받고 하기 위한 클래스 이다.   
   (3) 다른 사람과 연결하기 위한 Dialog를 생성한다.  
     
   (4) 버튼에 대한 BN\_CLIKED 이벤트와 도구상자에 대해 변수를 추가 시켜준다.   
   (5) 접속 버튼이 클릭되면 IP와 접속 버튼은 비활성화 되도록 하고 요청을 받은 사람이라면 게임시작이 비활성화 되도록 구현 한다.  
   (6) 게임 시작 버튼이 눌리면 Dialog가 꺼지고 게임이 시작되고, 대전 종료를 누르면 상대방에게 네트워크가 끊겼다고 메시지를 보낸다.
4. 네트워크를 통한 대전 모드 구현  
   (1) 계획한 내용 **5.**와 같이 메뉴를 추가한다.  
   (2) 게임을 하고 있는 중에는 대전만 활성화 될 수 있도록 한다.

# **최종보고서 본문**

최종보고서 작성

**최종결과물에 대하여 3장 이후 여러 장에 걸쳐서 명확하게 기술한다.**

# **자기평가**

자기평가작성

**여러 장에 걸쳐서 최종결과물에 대하여 기술한 다음 최종적으로는 본 프로젝트를 수행한 자기 평가를 기술한다. 최종 결과물에 대하여 다양한 준거를 적용하여 팀원 스스로 결과물의 사용 가능성 여부를 종합 판단한다.**