|  |
| --- |
|  |
| **MFC 프로젝트**  결과보고서 |
|  |

|  |  |
| --- | --- |
| **프로젝트명** | **MFC Warrior** |

|  |  |  |
| --- | --- | --- |
| **참여인원** | **20150235** | 박상순 |
| **20150250** | 신용구 |
| **20150291** | 하현수 |
|  |  |

**1. 프로젝트 개요 및 필요성**

* 1. **프로젝트 개요**

간단한 게임을 즐기면서 MFC에 대한 거부감을 줄이고, MFC 라이브러리 함수 활용법을 익힐 수 있다.

**1-2 주제 선정 이유**

어떤 프로젝트를 만드는 것이 좋을지 고민을 해보다가 제일 우리에게 친숙한 프로그램이자 수업시간에 주로 배운 화면출력을 최대한 활용할 수 있는 게임 구현을 주제로 결정했다. 주제가 결정된 후 구체적으로 어떤 게임을 만들지 여러 가지 게임을 검색해보며 직접 게임을 해보았지만 마땅히 좋은 게임은 나오지 않았으며, MFC만을 이용하여 만든 게임은 재미나, 퀄리티면에서 문제가 많아 차라리 재미를 조금 줄이더라도 교육적인 요소를 추가하자는 아이디어가 나와, 남녀노소 좋아하는 게임이라는 주제를 활용하여 MFC를 배울 수 있다면 그 자체로도 의미가 있을 것 같아서 주제를 선정하게 되었다.

**1-3 프로젝트의 중요성 및 필요성**

MFC는 C++기반이지만 대부분의 경우 C++과는 완전히 다른 라이브러리와 자료구조를 사용한다. 이로 인해 MFC를 처음 접하는 사람들은 모두 당황하고는 한다. 하지만 MFC의 라이브러리는 매우 간단하고 편하게 작성되어 있어, 사용법만 익히면 누구나 쉽게 사용 할 수 있다. 이를 본 프로젝트를 통해 간단히 익히면서, MFC에 대한 거부감을 줄이는 것 만으로도 MFC를 배우려는 사람들에겐 큰 도움이 될 수 있을 것이다.

**2. 프로젝트 개발 목표 및 내용**

**2-1 최종 개발목표**

게임을 단계별로 자주 쓰고 필요하다고 생각되는 함수를 하나 익힐 수 있도록 8 단계로 나누어 8가지의 함수를 함수원형을 주고 들어가는 인자들도 주어서 직접 함수를 작성하도록 하여 게임을 하는 사람들이 함수를 직관적이고 쉽게 익힐 수 있도록 하는 것이 목표이다.

**2-2 개발 개요**

MFC Warrior는 위에서 언급한 바와 같이 학습용 게임 프로젝트이다.

프로그램은 총 8스테이지로 나누어져 있고 기본적인 MFC 라이브러리 사용을 익히는것을 목표로 한다.

MFC에서 실제 빈번히 사용되는 예제를 이용하여, 코드 구현을 통해 스테이지를 진행하게 함으로써, 흥미를 이끌어 내었으며 문제에 함수의 원형을 주어주면서 실제 사용 방법을 재밌고 간단하게 익힐 수 있도록 했다.

User가 예제를 보고 정답코드를 작성하여 RUN버튼을 누르면 InputView클래스에서 코드를 비교하여 옳은지 아닌지 알려주게 된다.

MFC의 특성상 실행되어 있는 exe를 실시간으로 바꾸는 것이 안 되기 때문에 예제코드에 대한 답안을 미리 입력하여놓고 정답과 비교하여 실행하는 방식으로 구현했다.

만일 2번 이상 틀리게 될 경우 HINT 버튼을 띄워 답을 출력해 공부할 수 있도록 작성했다.

**2-3 Stage 정보**

Stage 1 : AfxMessageBox() 함수를 이용한 string 출력

Stage 2 : DC와 BitBlt() 함수를 사용하여 비트맵 출력

Stage 3 : DC와 Rectangle() 함수를 이용하여 사각형 출력

Stage 4 : DC와 LineTo() 함수를 이용하여 선 출력

Stage 5 : GetCurrentTime() 함수를 이용하여 배열에 삽입

Stage 6 : InsertAt() 함수를 이용하여 배열에 삽입

Stage 7 : CombineRgn() 함수를 이용하여 배열에 삽입

Stage 8 : CBrush를 이용하여 붉은색 브러시 생성

**2-4 개발 내용**

어디까지 입력되고 있는지를 더 잘 보여주기 위해 삽입할 때 커서를 출력해 주었다.

함수별로 자세히 보면 저희는 기존에 있는 View를 3등분하여 3개의 View(CCodeView, CInputView, COutputView)를 생성하여 각각 다른 내용을 출력하도록 했다.

분할하기위해 CMySplitterWnd 클래스를 작성하여 한 뷰에 3개의 뷰를 추가적으로 띄워 각각을 다르게 제어 할 수 있도록 해주었으며 뷰의 크기를 고정하지 않아 필요에 따라 늘리고 줄일 수 있는 효과를 낼 수 있도록 했다.

**CCodeView**

CCodeView는 예제코드를 출력해 준다. 예제코드는 실제 Visual Studio에서 사용하는 RGB값을 추출하여 특정 문자열 하이라이트(ex: void, CChildView,..etc) 기능을 추가 하였으며 실제 코드와 똑같이 구현했다. 각 단계를 실행하면 위 Stage에서 필요한 인자 등 모든 것들을 선언해놓았다. 집중력을 높이기 위해 배경색은 밝은 차콜을 사용했다.

**CInputView**

CInputView에서는 코드를 입력 받는다. 입력한 값을 실시간으로 View에 출력해주며 어디까지 입력이 되었는지 커서를 넣어 알기 쉽게 했다. 엔터를 누를 경우에는 RUN버튼을 누른 것과 같은 효과를 내도록 구현하여 편리성을 더했다. CInputView에서 입력받은 문자열을 답과 비교해 옳은지 아닌지 판단해주어 결과를 다른 클래스에 보내주는 역할을 한다. 코드를 입력하는 개인의 스타일이 다를 수 있으므로 공백 정보를 문자열 내에서 무시하도록 하여, 공백, 괄호를 넣는 방식이 다르더라도 정답 처리하도록 구현했다. 또한 CodeView와의 구분을 위해 이 뷰의 배경색으로는 밝은 회색을 사용했다.

**COutputView**

COutputView에서는 현재 단계의 미션과 문제, 함수 원형을 보여준다. 입력한 코드가 옳을 경우 스토리를 진행하기 위해 모든 움직임을 비트맵으로 관리하여 매 스테이지마다 실제 코드가 실행된 것 과 같은 효과를 내기 위해 최대한 비슷한 출력을 할 수 있도록 해주었다. RUN과 HINT, START버튼을 만들어 상황에 따라 출력하고 숨기는 작업을 한다. 또한 각 버튼이 눌렸을 경우 그 메시지를 처리한다. 정답을 입력하여 캐릭터가 맵 끝까지 도달하면 Stage변수를 늘려준 뒤 Invalidate하여 다음단계로 넘어간다.

**2-5 단계별 세부 설명**

Stage 0 : Start Button을 누른다.

Stage 1 : AfxMessageBox(\_T(\"앞으로가\")); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 앞으로가 라는 메시지박스를 출력해주며 앞으로 이동하여 다음 스테이지로 넘어간다

Stage 2 : dc.BitBlt(x, y, bi.bmWidth, bi.bmHeight, &dcmem, 0, 0, bi.bmWidth, bi.bmHeight, SRCCOPY);를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 캐릭터의 손에 칼이 생겨 가운데에 있는 나무를 베고 지나 다음스테이지로 이동한다.

Stage 3 : dc.Rectangle(x-25,y-50,x+25,y+50); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 손에 방패를 생성해 주고 앞으로 이동하여 다음 스테이지로 넘어간다.

Stage 4 : dc.LineTo(700,100); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 낭떠러지에 길을 생성하여 주고 앞으로 이동하여 다음 스테이지로 넘어간다.

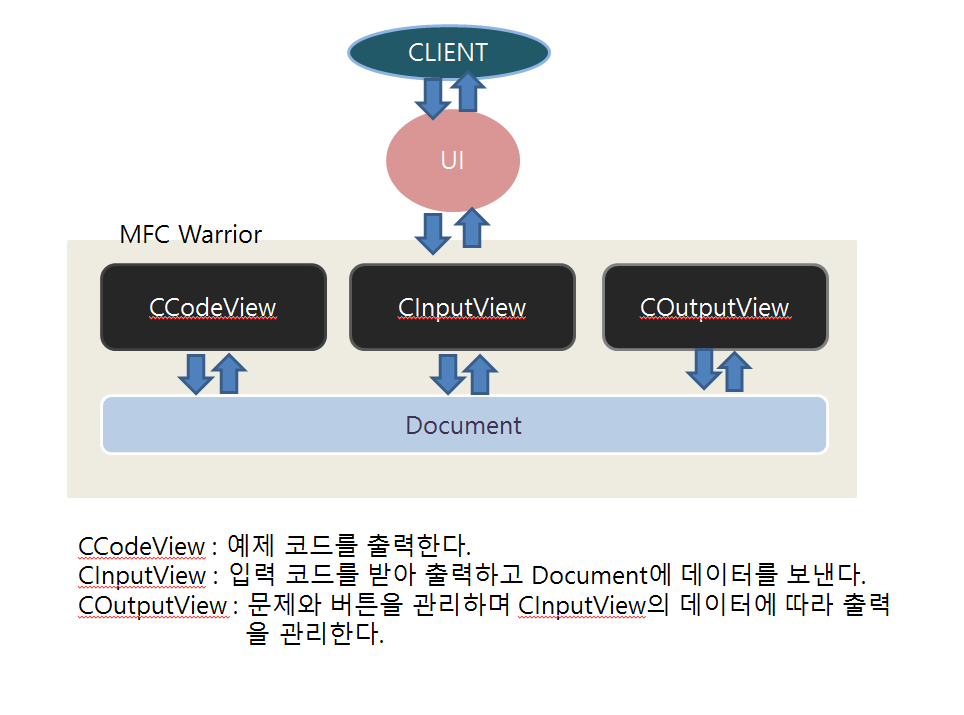
Stage 5 : time=CTime::GetCurrentTime(); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 현재시간을 메시지로 출력하여 주고 앞으로 이동하여 다음 스테이지로 넘어간다.

Stage 6 : password.InsertAt(3,2016); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 앞에 막혀있는 문이 열리며 다음 스테이지로 넘어갈 수 있다.

Stage 7 : new.CombineRgn(&original,&circle,RGN\_OR); OR new.CombineRgn(&circle,&original,RGN\_OR); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 조각나 있는 마법진을 완벽한 마법진으로 바꾸어 주며 앞으로 이동하여 다음 스테이지로 넘어간다.

Stage 8 : CBrush brush(RGB(255,0,0)); 를 제일 아래 뷰를 클릭한 후 입력한다. 그 후 왼쪽 위에 있는 RUN버튼을 눌러 실행시킨다. 만일 2번 이상 틀릴 경우 맨 위 뷰의 왼쪽 아래에 있는 HINT버튼을 눌러 답을 볼 수 있다. 옳은 경우에는 불덩어리가 앞으로 날아가 앞에있는 몬스터를 처치한 후 앞으로 이동하여 다음 스테이지로 넘어간다.

**2– 6 프로그램 구성도**



**3. 프로젝트의 기대 효과**

이 프로그램을 이용하여 MFC를 처음 배우는 사람, 혹은 프로그래밍을 처음 배우는 학생일 지라도 조금만 공부하면 손쉽게 즐길 수 있어 흥미를 유발 할 수 있다. 또한 스스로 작성해보고 틀리는 것을 반복함으로써 암기하기 위해 노력하지 않아도 스스로 터득하고 사용할 수 있게 된다. 이리하여 함수의 이해도를 높이고 자신이 아는 것을 점검할 수 있다.

**4. 역할 및 평가표**

|  |  |  |
| --- | --- | --- |
| 이름 | 역할 (각각 %는 전체에 대한 기여도) | 비율 |
| 박상순 | 리소스 제작, 관리(80%), 내용구현(20%) | 33.33/100 % |
| 신용구 | 리소스 제작, 관리(10%), 내용구현(60%), 디버깅(30%) | 33.33/100 % |
| 하현수 | 리소스 제작, 관리(10%), 내용구현(60%), 디버깅(30%) | 33.33/100 % |