소프트웨어 프로젝트II

**Adventure Design Project**

**보고서**

20181575 구다희

20181582 김명환

1. AD 프로젝트 주제

지뢰찾기 게임(Minesweeper)

1. 소프트웨어 요구사항 명세(**S**oftware **R**equirement **S**pecification)
   1. 기능적 요구사항
      1. 다양한 환경(View. 이를테면, CLI와 GUI)에서 사용할 수 있도록 디자인 패턴(MVC pattern과 Observer pattern)을 적용하여 코드의 재사용성을 보장한다.
      2. 게임 시작 전, 사용자가 게임 난이도(배열의 크기)를 선택할 수 있어야 한다.
      3. 지뢰찾기 배열 크기 선택지로, 초급(8x8), 중급(12x12), 고급(16x16) 세 가지를 제공한다.
      4. 지뢰찾기 난이도(배열 크기) Default는 고급으로 설정한다.
      5. 사용자가 배열 내 지뢰의 개수를 선택할 수 있어야 한다.
      6. 사용자가 지뢰 개수 범위를 지정할 수 있어야 한다. (1~(전체 크기-1))
      7. 지뢰 위치는 임의로(난수 발생적으로) 산출하여야 한다.
      8. 임의의 위치 추출 시 중복을 방지하여, 선택한 개수보다 적은 지뢰 개수가 적용되지 않도록 한다.
      9. 게임 옵션 설정 전(지뢰 개수 선택 전)에는 게임을 진행할 수 없다.
      10. 배열 크기에 따른 지뢰 개수를 추천할 수 있어야 한다. (확률 1/5~1/8 사이의 값)
      11. 선택한 지뢰 개수를 가시적으로 제시한다.
      12. 한 번 선택한 구역은 다시 선택할 수 없다.
      13. 기존 지뢰찾기 게임에서 제공하였던 깃발(Flag) 기능을 동일하게 제공한다.
      14. 깃발은 전체 게임 로직과는 별개로 메모 용도로만 사용한다.

(즉, 깃발 추가가 아닌 구역 선택으로 게임을 진행한다.)

* + 1. 깃발 선택 시, 옵션에 따라 다른 로직을 적용하여 선택 결과를 출력한다.
       1. 깃발이 위치된 구역을 선택(왼쪽 마우스 클릭) 시 선택한 결과를 출력한다.
       2. 깃발 재선택(오른쪽 마우스 클릭) 시 위치된 깃발을 삭제한다.
    2. 선택한 구역이 황무지(인접 지뢰 개수가 0인 구역)일 경우, 인접 지뢰 개수가 1이상인 구역까지 확장 선택하여 출력한다.
    3. 확장 출력 시 깃발이 위치된 구역을 포함할 경우, 해당 구역은 12-(i)를 적용하여(즉, 왼쪽 마우스 클릭한 경우로 간주하여) 출력한다.
    4. 사용하는 깃발의 개수는 지뢰의 개수를 초과할 수 없다.
    5. 지뢰의 개수, 남은 구역, 사용한 깃발의 개수와 같은 동적인 값들을 변경 시마다 출력한다.
    6. 지뢰 영역을 선택하면 게임이 종료한다. (실패)
    7. 지뢰 영역을 제외한 모든 구역을 선택할 경우 게임이 종료한다. (성공)
    8. 게임이 종료되면 지뢰의 위치와 함께 종료 메시지를 출력한다.
    9. 종료 메시지는 게임 성공 여부에 따라 다르게 출력한다.
    10. 종료 결과에 따라 게임 성공 여부에 따라 다른 모양 및 글씨색으로 출력한다.
    11. 게임이 끝난 이후에는 사용자가 더 이상 게임을 진행할 수 없어야 한다.
    12. 게임 진행 부분 외에 종료, 포기, 옵션 초기화, 재시작 총 4가지 메뉴를 제공한다.
    13. 포기 메뉴 선택 시 정답을 출력한다.
    14. 옵션 초기화 메뉴 선택 시 배열 크기부터 다시 선택하도록 한다.
    15. 재시작 메뉴 선택 시 배열 크기는 이전 게임의 설정값을 이용하며, 지뢰 개수부터 선택한다. (지뢰 개수 선택 시, 지뢰 위치는 난수 발생적으로 산출)
    16. 재시작 혹은 옵션 초기화 메뉴 선택 시, 옵션 선택 이후의 로직은 처음 게임 실행 시와 동일하게 진행되어야 한다.
    17. 디자인 패턴 구현을 위해, observable과 observer 모듈을 Model과 View 클래스에서 각각 상속받아 사용한다.
    18. 클릭한 구역(버튼)은 남은 구역과의 구별을 위해 구역 배경을 회색으로 변경한다.
    19. 인접한 지뢰의 개수에 따라 각기 다른 글씨색을 출력한다.
  1. 비기능적 요구사항
     1. 본 소프트웨어의 구현에는 Python과 Pyqt5를 이용한다.
     2. 게임 실행 시 출력되는 모든 내용에 있어서, 언어는 영어로 한정한다. 단, 게임 코드는 내용 파악을 위해 한글 주석 사용도 허용한다.
     3. 예외처리를 위해 게임 로직을 기능, 단계별로 나누어 진행한다.
     4. 전체 시스템에 있어서, 게임 진행 도중 오류가 발생하지 않아야 한다.
     5. 예외가 발생할 수 있는 상황에 대해, 최대한 미연에 방지할 수 있도록 처리한다.
        1. 일련의 예로, 예외가 발생할 수 있는 상황은 다음과 같은 사항이 있다.
           1. 개수 선택 시 : 입력(Input) 타입과 범위 🡺 QComboBox로 내용 한정
           2. 난이도(배열의 크기) 선택 시 : 입력(Input) 타입과 범위(범위는 계산 속도와 효율성, 즉 게임 로직과 특히 연관 된다. model.py – Model 클래스의 guess() 메소드 참조) 🡺 QMessageBox의 Button들로 구현
           3. 구역 선택 시 : 배열 초과 범위 항목 선택(Index Out of Range) 🡺 구역에 해당하는 값들만 버튼으로 구현하여 가시화
     6. 게임 사용 설명서(도움말)는 별도로 작성하지 않되, 사용자가 게임 진행 순서를 자연스럽게 인식할 수 있도록 부가 기능을 추가한다.
        1. 게임 진행 순서 : 게임 난이도 선택 (-> 지뢰 개수 추천) -> 지뢰 개수 선택 -> 게임 진행 -> 게임 종료
        2. 게임 진행 중 메뉴는 언제든 선택 가능하며, 선택하는 메뉴에 따라 게임 진행 순서는 일부 순환, 생략, 반복될 수 있다. (역순 진행은 절대 불가)
        3. 게임 난이도는 메인 화면 실행 전 QMessageBox를 통해 사용자에게 선택지를 제공한다.
        4. 지뢰 개수는 QComboBox를 통해 선택지를 제공한다. QComboBox에서 개수 선택 후 선택 버튼을 클릭하기 전에는 구역을 선택할 수 없어 게임 진행이 불가능하다.
        5. 모든 옵션 선택 후, 본격적으로 게임을 진행한다.
        6. 게임 종료 후에는 더 이상 구역과 옵션을 선택할 수 없다. (메뉴만 선택 가능)
        7. 위와 같은 예처럼, 진행 순서의 인식을 위해 반드시 해당 순서의 내용을 거치도록 조작한다.
  2. 사용자 인터페이스 요구사항
     1. 해당 게임 진행을 위해 필요한 기능별 항목은 다음과 같다.
        1. 전체 게임 윈도우 창 (타이틀 및 몸체)
        2. 난이도 선택창
        3. 개수 선택창
        4. 지뢰 개수 출력창
        5. (지뢰찾기 메인화면) 구역 선택창
        6. 게임 진행 상태 표시창 : 난이도(배열 크기), 미확인 구역 개수, 깃발 개수
        7. 게임 종료 시 출력되는 메시지창
        8. 글씨 및 글씨색 설정을 위해 필요한 모듈들
     2. 이를 구현하기 위한 GUI 요소는 다음과 같다.
        1. QApplication : 이벤트 루프 진입과 게임 로직 진행을 위해 사용
        2. QWidget : 전체 게임 윈도우 창 구현을 위해 사용
        3. Qt : QLabel의 내용 맞춤(배치)에 사용
        4. QGridLayout : GUI(외적인 측면)와 계층(내부적 측면) 구조를 효율적, 효과적으로 구현한다. 특히 지뢰 배열에서 효과적!
        5. QComboBox : 지뢰 개수의 효율적 입력과 에러 방지를 위해 사용
        6. QMessageBox : 주목할 만한 내용을 사용자에게 알리거나 전체 게임 진행을 관리할 때 사용
        7. QToolButton : 정확히는 이를 상속받아 메소드를 오버라이딩한 button 클래스를 사용. 버튼 하나가 지뢰찾기의 각 구역을 1:1 대응 관계로 담당한다.
        8. QPushButton : 지뢰 개수 최종 선택을 사용자로부터 입력받을 때 사용
        9. QGroupBox : 게임 UI를 각 기능별로 분리 🡪 기능별 관리 용이(개발자 측면), 기능 파악의 편의(사용자 측면)를 위함
        10. QLayout : GUI 구성을 위해 사용
        11. QLabel : 상태 메시지를 출력하거나, 혹은 이 수치가 무엇을 의미하는지를 사용자에게 알리기 위해 사용
        12. QFont : 폰트 크기 조정에 사용
        13. QSizePolicy : 버튼 크기 조정에 사용

1. 소프트웨어 구조 설계(Architecture Design Description)

스크린샷이(가) 표시된 사진

자동 생성된 설명

|  |  |  |
| --- | --- | --- |
| 모듈 | 클래스 | 역할 |
| MineSweeper.py | MineSweeper | 게임 메인 드라이버 |
| model.py | Model | 지뢰찾기 게임의 핵심 로직 구현. 게임 데이터와 상태에 대한 모든 정보를 포함한다. |
| button.py | Button | 지뢰 배열의 버튼을 구현 |
| controller.py | Controller | Model과 View 클래스의 연결다리 역할. View로부터 정보를 받아 Model에서 사용하기 적합한 형태로 내용을 요청한다. |
| view.py | View | 사용자 인터페이스(GUI) 구현을 위한 PyQt5 QWidget 등을 포함. 지뢰찾기 게임의 외관을 담당 |

1. 구현 상세 설계(Detailed Design Description)

(모른척) 졸귀탱구리

1. 단위 테스트(Unit Test)
2. 주요 로직을 담당하는 model.py 모듈 테스트를 진행
3. 모든 메서드를 테스트 하는 것이 좋지만, 주요 기능만 테스트 하는 것도 가능하며, 오히려 시간을 절약할 수 있다.
4. testSetArray를 포함해 테스트의 의미가 없는 메서드는 testSetArray를 제외하고, 더 이상 Test case를 작성하지 않았다.

|  |  |
| --- | --- |
| Test case | 역할 |
| setUp | Test suite 초기 값 설정 |
| testGetStatus | 현재 게임의 상태를 notifly()하는 메서드의 3가지 모든 상태를 확인한다.. |
| testGuess | 빈 공간을 눌렀는지, 폭탄 공간을 눌렀는지 계산해주는 메서드의 2가지 상태를 확인한다. |
| testSetArray | 사용자의 입력 값에 따라 배열의 크기를 정해주는 메서드를 테스트한다. |

1. 통합 테스트(Integration Test)

이건.. 전체 실행 사진으로 때우면 되는 건가 ㅇㅅaㅇ

PPT에 넣을 사진들 넣으면 되겠군!

1. 테스트 보고서 (5번을 지우고 7번 한 개로 둬도 될 것 같은데?)

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | UTT | Description | Result |
| UT 1-1 | model.getStatus | 현재 게임 상태를 알려주는 기능 테스트 | Pass |
| UT 1-2 | model.guess | 주변에 지뢰가 있는지 | Pass |
| UT 1-3 | model.setArray | 배열의 크기를 설정하는 기능 테스트 | Pass |