2023년 2학기

프로그래밍과 문제해결

Assignment #4

담당교수: 윤은영
학번: 20230673
학과: 무은재학부
이름: 전재영

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Problem: 지뢰찾기

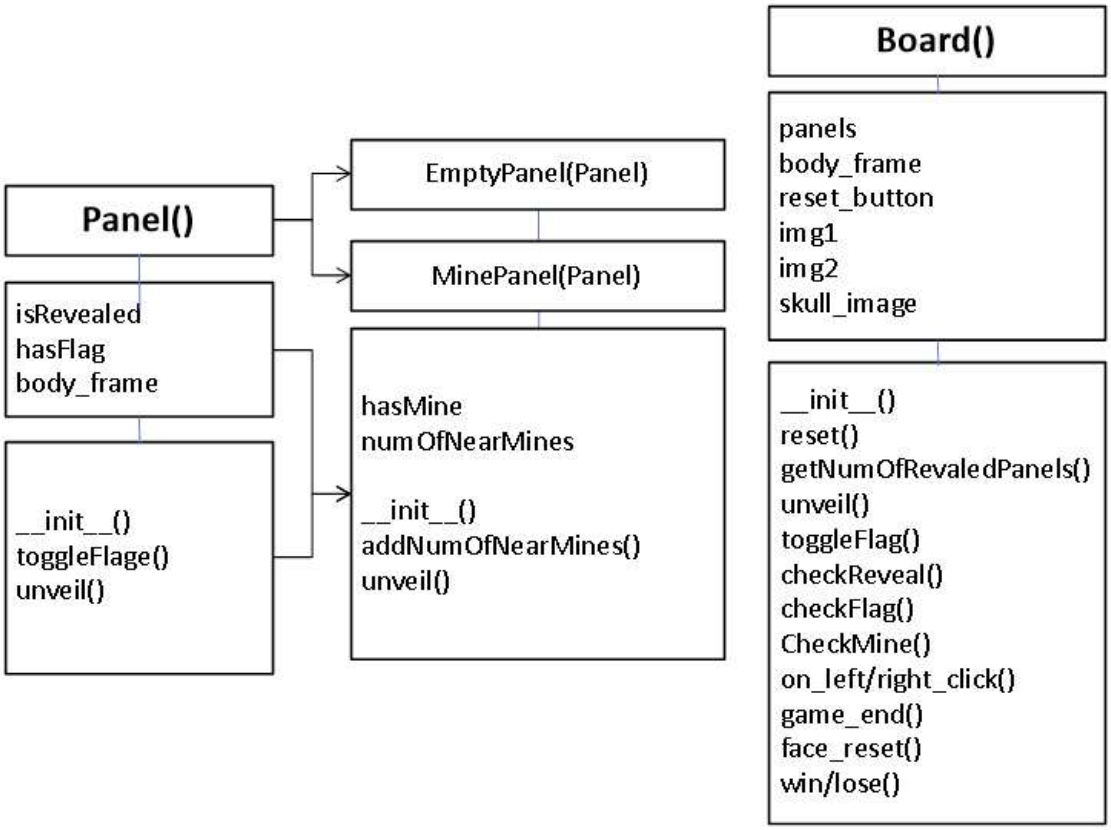
1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

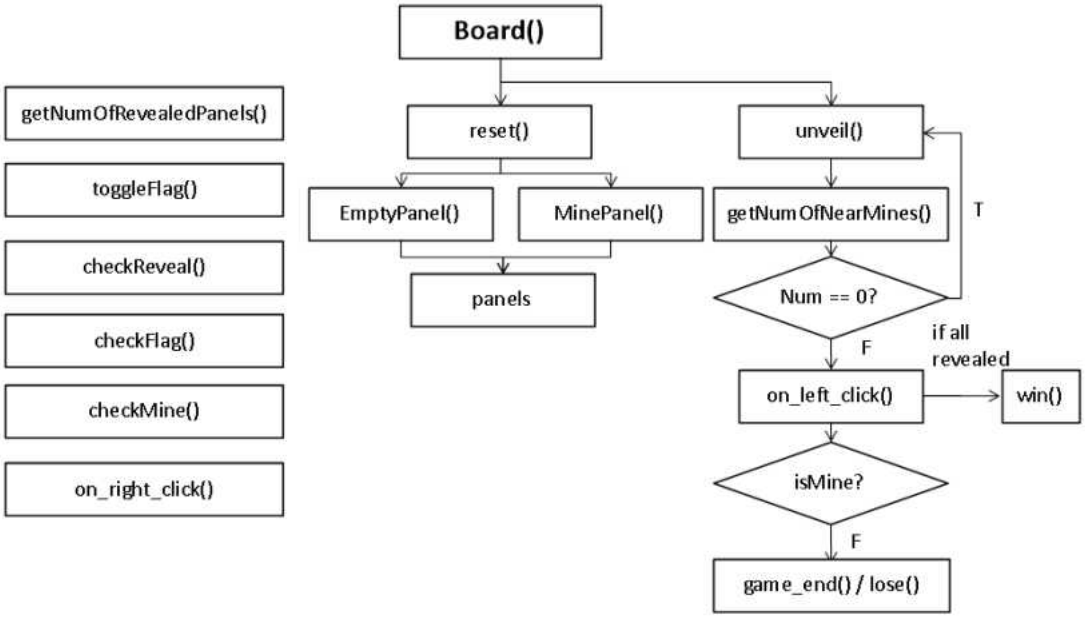
- model.py를 통해 게임에 필요한 패널, 보드 등을 구현하고, 지뢰찾기를 구현한다.
- app.py에서 tkinter를 사용해 gui를 구현한다.

2. 알고리즘

이 프로그램의 클래스 구조도는 다음과 같다.



이 프로그램의 메인이 되는 Board()의 flowchart는 다음과 같다.



지뢰 찾기에서 대표적인 리셋버튼과 패널을 드러내는 부분을 나타냈으며, 자세한 작동과정은 아래에서 설명한다.

이 프로그램의 Pseudo-algorithm 은 다음과 같다. 프로그램을 진행하는 model.py만 설명하겠다.

```
Panel : {isRevealed = False, hasFlag = False}
-> toggleFlag() : !hasFlag
-> unveil() : isRevealed = True
EmptyPanel(Panel) : {hasMine = False, numOfNearMines = 0}
-> addNumOfNearMines() : numOfNearMines += 1
-> unveil() : unveil Panel, return this numOfNearMines
MinePanel(Panel) : {hasMine = False, numOfNearMines = 0}
-> addNumOfNearMines() : numOfNearMines += 1
-> unveil() : unveil Panel, return -1
Board() : (panels = [] )
-> reset() :
        buttons, mines = [[]]
        all clear panels
        ramdomly add 1 in mines, other is 0
        for x : for y : if mines[y][x] == 1 : add MinePanel() to panels
                        else : add EmptyPanel() to panels
        make button, and add to buttons,
        for x : for y: for (x-1, x, x+1) and (y-1, y, y+1) :
```

if mines[a][b] is mine : panels[a][b]'s numOfNearMines += 1
-> left_click() : reveal button, if it's numOfNearMines == 0:
left_click() for around 3 * 3
if it was mine, reveal all button, and game lose.
if reveal all button except for mine, and game win
-> right_click() : is hasFlag -> make Flag, is !hasFlag -> delete Flag

3. 프로그램 구조 및 설명

먼저 프로그램을 이루는 클래스들과 변수들, 함수들을 설명하고, 구조를 설명하겠습니다.
++ 표시가 되어있는 변수나 함수는 추가한 내용입니다.

클래스 :
Panel() : 지뢰찾기의 각 칸에 해당하는 클래스로, 상위 클래스에 해당합니다.
- isRevealed : panel이 밝혀진 상태인지를 나타내는 bool 변수입니다.
- hasFlag : panel이 flag를 보유하고 있는지를 나타내는 bool 변수입니다.
- ++ body_frame : model.py 파일에서 버튼을 생성하고 관리하기 위해, app에서 버튼이 생성된 body_frame을 받아옵니다.

-> toggleFlag(self) : 해당 panel의 hasFlag를 toggle합니다.
-> unveil(self) : 해당 panel을 밝혀진 상태로 변경합니다.

EmptyPanel(Panel) : Panel() 클래스를 상속받아 만들어지며, 지뢰가 없는 panel을 나타냅니다.
- numOfNearMines : 주변의 Panel이 지뢰를 갖고 있는 수입니다.
- ++ hasMine : 지뢰를 갖고 있는지 나타내는 값입니다. EmptyPanel에서는 Fasle의 값을 갖습니다.

-> addNumOfNearMines(self): 해당 panel의 numOfNearMines 값을 1 증가시킵니다.
-> unveil(self): # 부모 class의 unveil을 실행하고, numOfNearMines를 리턴합니다.

MinePanel(Panel): EmptyPanel()과 유사하며, hasMine이 True라는 것을 제외하면 작동 방식은 같습니다.
- numOfNearMines : 주변의 Panel이 지뢰를 갖고 있는 수입니다.
- ++ hasMine : 지뢰를 갖고 있는지 나타내는 값입니다. MinePanel()에서는 True의 값을 갖습니다.

-> addNumOfNearMines(self): 여기서는 사용하지 않지만, EmptyPanel과의 작동을 동일하게 하기 위해 pass로만 구현만 해둡니다.
-> unveil(self): # 부모 class의 unveil을 실행하고, -1을 리턴합니다.

Board() : 지뢰찾기 게임에서 모든 게임 부분을 구현한 클래스입니다.
- panels : Panel 클래스들을 담는 2차원 리스트입니다. 이 패널들을 인덱스로 접근하면서 게임의 연산을 수행합니다.

- ++ body_frame : 버튼에 대한 함수처리를 위한 app.py에서 만들어진 body_frame입니다.
- ++ reset_button : app.py에서 만들어진 reset_button에 대한 함수처리를 위해 받아온 변수입니다.
- ++ img1, img2, skull_image : reset_button의 함수에서 사용되는 이미지로, 각각 기본, 승리, 패배의 상태를 나타냅니다.

-> reset(self, numMine, height, width): 보드판을 초기화하는 함수입니다. 먼저 reset_button의 이미지를 기본으로 바꾸고, 기존에 있던 보드를 삭제합니다. 그후, 받은 height와 width에 기반하여 지뢰 생성, Panel 생성, button 생성, numOfNearMines 기록 등의 초기화를 진행합니다.
-> getNumOfRevealedPanels(self) : 현재 밝혀져 있는 panel의 개수를 반환합니다.
-> unveil(self, y, x): y행, x열의 Panel을 드러냅니다. 만약, 패널의 numOfNearMines 이 0이었을 때, 주변 8개의 Panel에도 재귀함수를 이용해 드러내는 작업을 실행합니다.
-> toggleFlag(self, y, x): 해당 위치의 flag를 토글합니다.
-> checkReveal(self, y, x): 해당 위치의 isRevealed를 리턴합니다.
-> checkFlag(self, y, x): 해당 위치의 hasFlag를 리턴합니다.
-> checkMine(self, y, x): 해당 위치의 hasMine를 리턴합니다.
-> getNumOfNearMines(self, y, x): 해당 위치의 numOfNearMines을 리턴합니다.
-> on_left_click(self, row, col): 해당 Button을 좌클릭 했을 때의 처리 함수입니다. 해당 위치의 Panel을 드러내는 작업을 수행하고, 만약 지뢰라면 game_end()를 통해 게임을 종료합니다. 이미 드러난 Panel에 대해서는 수행하지 않습니다. 모든 지뢰 이외의 패널을 드러냈을 때, win()을 실행하며, 게임을 승리합니다.
-> ++ on_right_click(self,row, col): 해당 Button을 우클릭했을 때의 처리 함수입니다. 만약 Flag가 놓여있지 않은 경우, Flag를 추가하며, 이미 존재하는 경우, Flag를 지웁니다. 이미 드러난 Panel에 대해서는 수행하지 않습니다.
-> ++ game_end(self): 게임을 패배해 종료되었을 때를 나타내며, 모든 Panel을 드러내고, lose()를 통해 reset_button을 해골 모양으로 바꿉니다.
-> ++ face_reset(self): Board를 초기화할 때, 이미지를 img1으로 설정합니다.
-> ++ win(self): 승리했을 때, reset_button의 이미지를 img2로 설정합니다.
-> ++ lose(self): 패배했을 때, reset_button의 이미지를 skull_image로 설정합니다.

변수 :
TITLE, BTN_WIDTH, BTN_HEIGHT, NUM_MINE : 프로그램의 제목, Panel의 개수, 지뢰의 개수를 나타내는 변수입니다. 난이도에 따라서 BTN_WIDTH, BTN_HEIGHT, NUM_MINE가 변경됩니다.
button_list : 버튼을 담는 2차원 리스트입니다.
mylist : 지뢰의 위치를 나타내는 2차원 리스트입니다. 지뢰의 경우 해당 위치에 1로 기록이 되어 있으며, 아닌 경우 0으로 기록되어 있습니다. 지뢰를 탐색하는 경우, mylist에서 인덱스로 지뢰를 탐색한 후, 이를 button_list나 Board의 panels에 이용해 수행하는데에 도움을 줍니다.

/// 여기부터는 app.py에 대한 설명입니다. ///

클래스 :

- App(tk.Frame) : 클래스 형태로 구현된 Frame입니다. Board에서 처리하는데에 필요한 body_frame, reset_button을 전달하기 위해 header_frame, body_frame이 정의되어 있으며, 이외의 gui는 main()에서 구현되어 있습니다.

함수 :

- reset_button_left(event): reset_button을 좌클릭 했을 때 실행하는 함수로, Board를 리셋합니다.
- easy_mode() : BTN_WIDTH = 10, BTN_HEIGHT = 10, NUM_MINE = 10으로 설정하고, Board를 리셋합니다.
- normal_mode() : BTN_WIDTH = 15, BTN_HEIGHT = 15, NUM_MINE = 30으로 설정하고, Board를 리셋합니다.
- hard_mode() : BTN_WIDTH = 20, BTN_HEIGHT = 20, NUM_MINE = 50으로 설정하고, Board를 리셋합니다.
- main() : App 클래스의 instance를 만들고, App 클래스에서 만들지 않은 메뉴바를 만들고, 메인루프를 시작합니다.

구조도에 대한 설명은 다음과 같습니다.

1) 프로그램 실행

- 프로그램이 실행되면 app.py에 있는 App() 클래스를 통해 기본적인 GUI를 만듭니다. App() 클래스에서는 기본적으로 head_frame과 여기에 위치한 reset_button, body_frame을 만들며, 이외의 레벨바와 같은 기능은 app.py의 main()에 정의되어 있습니다.
- App()에서 model.py에 있는 Board를 만드는데, 이를 생성할 때, 버튼에 대한 함수를 처리하기 위해 만든 reset_button과 body_frame을 매개변수로 넘겨서 생성합니다.

2) Board()의 생성과 그 처리

- Board를 생성하면 reset(self, numMine, height, width)을 통해 초기화하는 과정을 진행합니다. 먼저, reset_button의 이미지를 초기 상태로 되돌리고, 게임을 재시작하는 경우를 대비해 기존의 버튼들을 삭제하는 과정을 거칩니다. 지뢰의 위치를 나타내는 mylist, 패널을 담는 self.panels, 버튼을 담는 button_list를 초기화하고, 랜덤으로 지뢰를 생성하는 과정을 거칩니다.
- mylist에 지뢰를 1로 기록하고, 그 이외는 0으로 기록합니다. mylist의 인덱스에 대응시켜서 panels에 Panel을 넣는데, 만약 1인 경우(지뢰)에는 MinePanel()을 생성해서 넣고, 아닌 경우는 Panel() Empty()를 넣습니다.
- button_list에는 좌클릭을 했을 때, Panel을 unveil() 하고, 우클릭 했을 때 플래그를 세우는 함수 on_right_click()의 기능을 있는 button을 생성해 넣습니다.
- 그후, 모든 Panel에 대해 주변 3 x 3 패널을 탐색해 지뢰가 존재하는 경우 addNumOfNearMines()를 이용해 주변에 있는 지뢰 개수를 기록합니다.

3) 게임의 처리 (좌클릭)

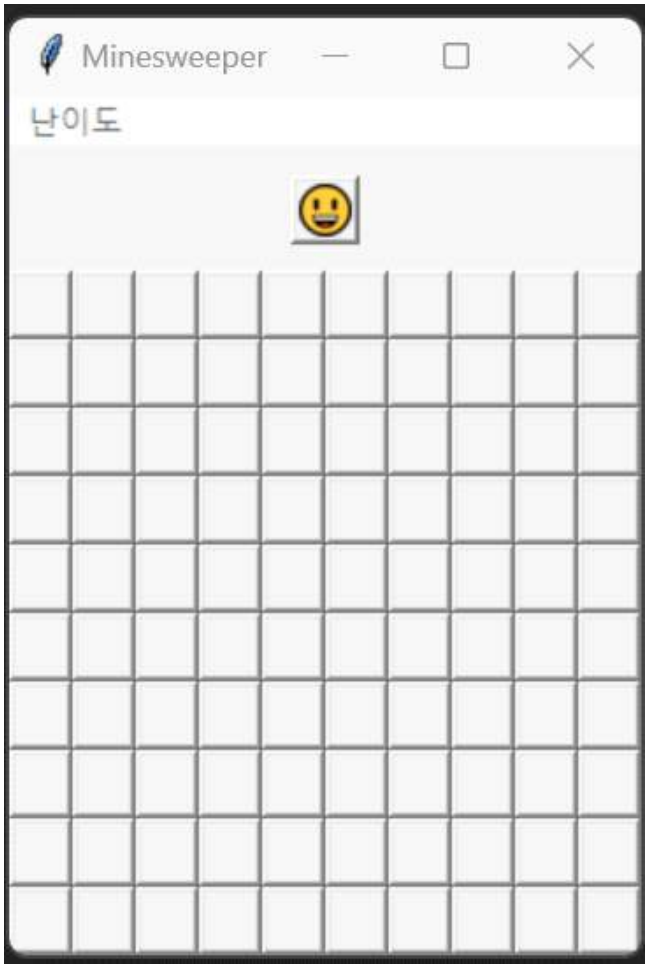
- 게임에서 Panel(button)을 좌클릭 하면 그 위치를 드러냅니다. Board()의 unveil(self, y, x) 함수를 이용해 그 on_left_click()을 통해 위치를 드러내며, 만약 그 위치의 Panel의 numOf-NearMines가 0일 경우에는, 자신을 제외한 주변 3 * 3의 Panel에도 unveil()을 재귀적으로 수

행합니다.
- checkReveal()을 통해 탐색해 이미 드러난 Panel에 대해서는 아무런 작업도 수행하지 않습니다.
- on_left_click()에서는 버튼을 비활성화 상태로 바꾸고, 그 패널을 드러냅니다. 만약에 그 패널이 지뢰가 존재하는 패널이었을 경우, game_end()를 수행해 모든 패널을 드러내고 lose()를 통해 맨 위의 reset_button의 이미지를 해골 모양으로 변경합니다.
- 만약 지뢰 이외의 모든 패널을 드러냈을 때, win()을 수행하며, 맨 위의 reset_button의 이미지를 선글라스 모양으로 변경합니다.

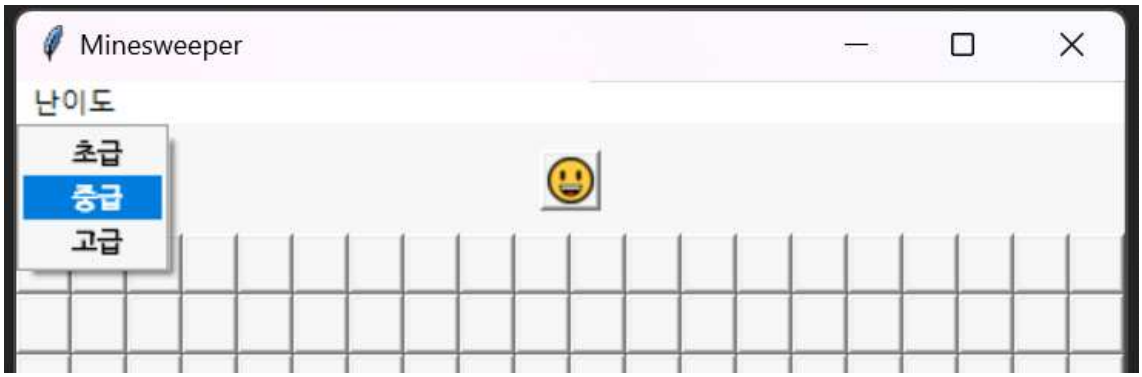
4) 게임의 처리(우클릭)
- 우클릭의 경우, on_right_click()을 실행해 해당 패널을 탐색합니다. 만약 checkReveal()을 통해 이미 드러나 있는 경우 아무런 작업도 수행하지 않습니다. checkFlag()를 통해 isFlag가 False인 경우에 패널에 isFlag가 True로 만들고, Flag를 버튼에 표시합니다. 반대로, isFlag가 True인 경우에는 이미 존재하는 Flag를 없애고, isFlag를 False로 만듭니다.

5) GUI
- app.py에서는 main()에 메뉴바를 만들어 난이도를 선택할 수 있습니다. 초급의 경우, 10 * 10에 지뢰가 10개, 중급의 경우 15 * 15에 30개, 고급의 경우 20 * 20에 지뢰가 50개로 설정합니다.
- app에서 board의 reset()을 통해 시작하면 초급 난이도의 버튼을 생성할 수 있도록 합니다.
- 그 후, mainloop()를 통해 프로그램을 진행합니다.

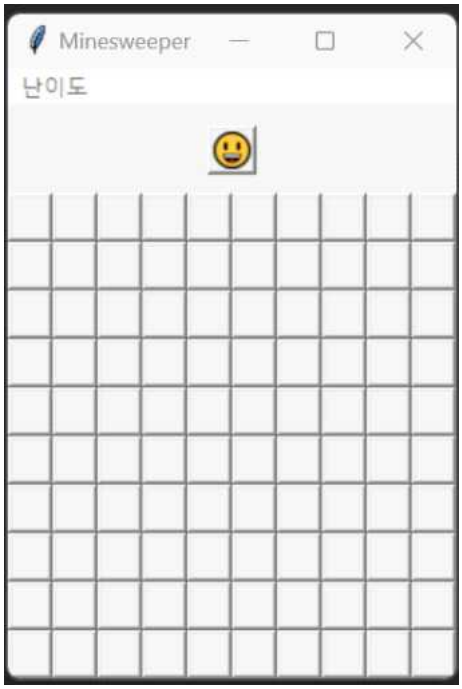
4. 프로그램 실행방법 및 예제
- vscode를 이용해 제작과 실행을 했습니다.



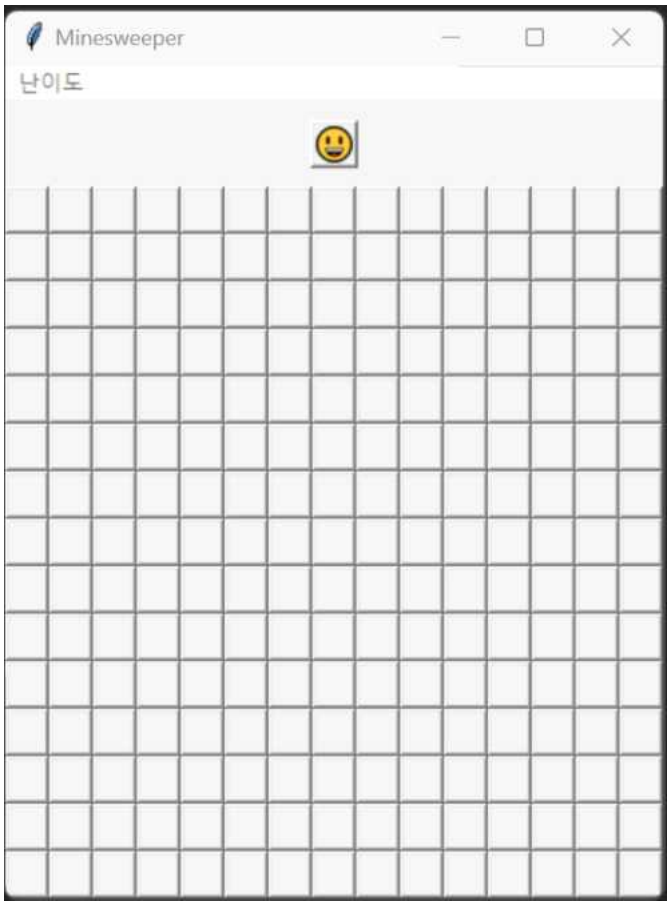
1. 프로그램을 시작하면 다음과 같이 10 * 10의 화면이 출력된다. 중간에 초기화할 수 있는 버튼이 있고, 왼쪽 위에 난이도를 조정할 수 있는 창이 있다.



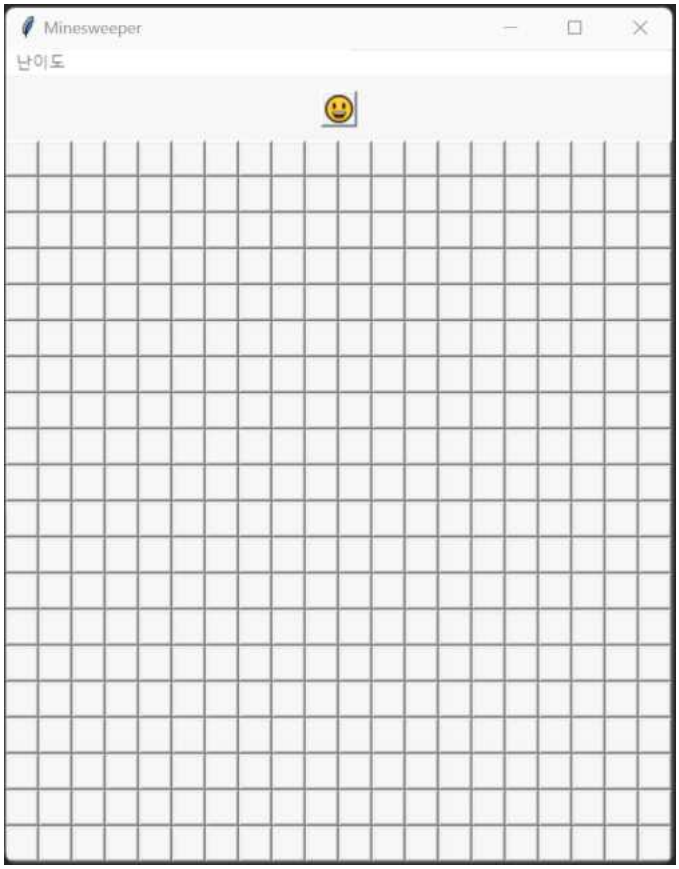
2-0. 난이도를 통해서 초급, 중급, 고급 중에서 하나를 고를 수 있다.



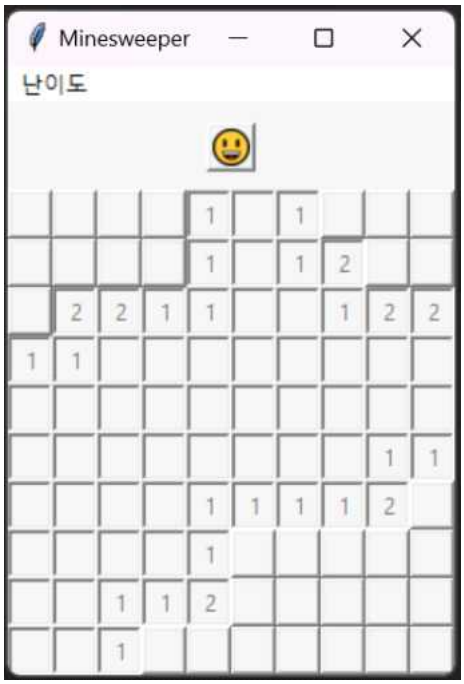
2-1. 초급을 고른 경우. (10 * 10, 지뢰 10개)



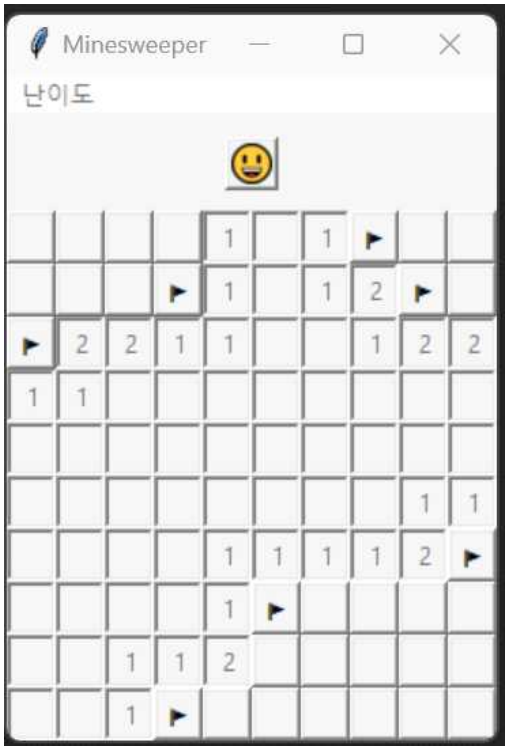
2-2. 중급을 고른 경우 (15 * 15, 지뢰 30개)



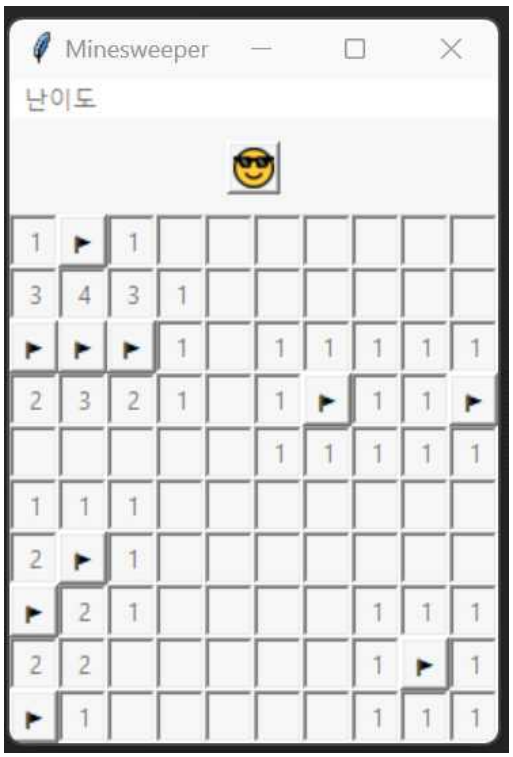
2-3 고급을 고른 경우 (20 * 20, 지뢰 50개)



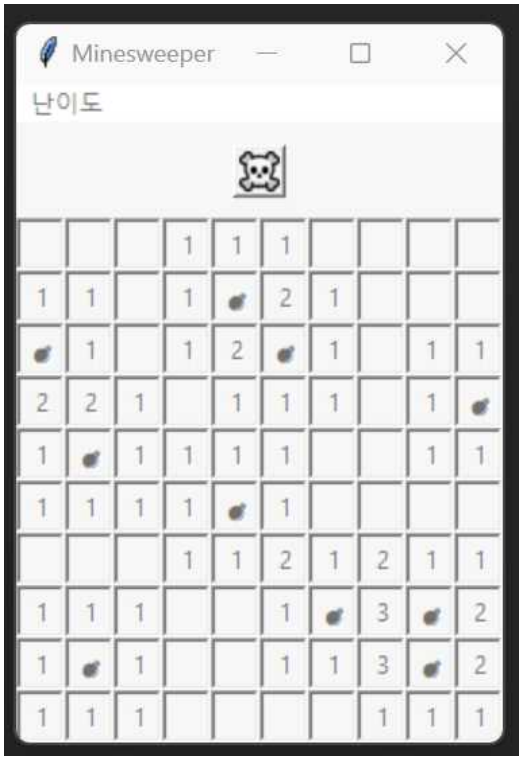
3. 만약 주변의 지뢰 개수가 0인 버튼을 눌렀을 경우, 주변에 지뢰가 존재하는 버튼까지 계속 해서 주변을 드러낸다.



4. 우클릭을 눌러 깃발을 통해 지뢰의 위치를 기록할 수 있다. 만약 토글된(이미 깃발이 있는) 버튼에 다시 우클릭을 할 경우, 깃발이 지워진다. (텍스트로 깃발을 표현했다.)



5. 만약 지뢰 이외의 모든 패널을 드러냈을 때, 유저는 승리하며, 상단의 스마일 버튼이 선글라스를 끼고 있는 모습으로 바뀐다.



6. 만약 지뢰가 있는 패널을 선택했을 때, 모든 패널을 드러내며, 상단의 스마일 버튼이 해골 표시로 바뀌게 된다.

5. 토론

- 지뢰의 위에 기록하는 깃발이나 폭탄의 경우, 버튼에 사진을 넣으면 버튼에 사진이 생기는 것이 아니라, 버튼 위에 그림의 모양을 한 아주 작은 버튼이 생기는 문제가 발생했다. 그래서 이미지 대신 텍스트의 형식으로 깃발과 폭탄을 표현하였고, 생기는 문제를 해결할 수 있었다.
- 처음에 과제를 진행할 때, 모델과 앱을 한 .py에 구현해 이를 분리하는 방법을 찾는 것이 어려웠다. 한 파일에 코딩을 할 때는, button에 관한 함수를 처리할 때, main()에서 이미 button이 생성되고, 함수를 불러오기 때문에 button의 존재성에 대해 생각할 필요가 없었는데, 앱과 모델을 분리하니, 클래스에서 button에 대한 함수를 처리하려 할 때, button의 생성보다 먼저 button을 참조해 실행할 수 없는 오류가 발생했다. 그래서, app.py에서 main()에 모든 frame을 구현하기 보다, frame을 클래스 형식으로 만들어 button과 frame을 생성하고, 이를 클래스에 전달해 참조 오류를 해결할 수 있었다.

6. 결론

- 본 과제를 통해 파이썬에서의 클래스를 만들고 구현하면서, OOP에 대한 개념을 익힐 수 있었다. 클래스를 만들고 상속하며, override에 대한 개념도 익힐 수 있었다. 여러 코드로, 게임으로 구현해 보면서 성취감도 얻을 수 있었던 시간이었던 것 같다.

7. 개선방향

- 코드가 심각하게 비효율적으로 구현되어, 중급~고급 정도에서는 보드를 생성할 때 눈에 보일 정도로 랙이 걸린다. 각 패널마다 주변에 있는 지뢰의 개수를 저장해야하기 때문에, 이를 한 패널당 주변 3*3의 패널을, 총 20 * 20번 탐색하는 것이 문제가 되었다고 생각한다. 만약 패널을 모두 만든 후, 지뢰를 넣어준 다음에, 지뢰의 위치 주변 3*3의 위치에 지뢰의 개수를 1개씩 올려주는 식으로 코드를 구현했으면 어땠을까 하는 아쉬움이 있다.

참고 : Lab11 - GUI