Minesweeper - dokumentacja

# Czym jest projekt?

Projekt ten jest implementacją Sapera – znanej gry z systemu Windows. Celem gracza jest odkrycie wszystkich pól niezawierających bomby. Gra kończy się w momencie odkrycia wszystkich pól bez bomby ( zwycięstwo ) , lub odkryciem pola z bombą ( przegrana ). Gracz do dyspozycji ma flagi, dzięki którym może zaznaczać gdzie jego zdaniem znajduje się bomba. Projekt został zaimplementowany z pomocą biblioteki pygame.

# Jak uruchomić grę?

Grę uruchamia się przechodząc do katalogu z projektem, następnie wpisując w linii następującą komendę : „python main.py {poziom trudności}” lub „python3 main.py {poziom\_trudności}” , gdzie poziom trudności to liczba od 1 do 3

# Omówienie plików projektowych

W projekcie znajdują się następujące klasy:

- main.py – do uruchamiania programu

- minesweeper.py – obsługa ekranu gry

- board.py – obsługa planszy gry

- field.py – obsługa pojedyńczego pola na planszy

Oprócz tego znajdują się pliki pomocnicze np. Ikony pól czy pliki dźwiękowe odtwarzane w trakcie gry

* 1. Main.py

Najkrótsza z klas, pobiera argument z linii poleceń, ustawia odpowiednią planszę i rozpoczyna grę

* 1. Minesweeper.py

Tutaj dzieje się główna logika gry. To w tej klasie znajduje się obsługa biblioteki pygame, która generuje ekran gry i obsługuje dane wejściowe ( mysz ). Zawiera następujące metody :

Konstruktor – inicjalizuje grę, ładuje ikony

Run – główna pętla programu. Tutaj program wyświetla z pomocą innych metod klasy ekran gry

Load\_images – ładuje obrazki z wskazanego katalogu

Draw – rysowanie planszy na ekran, a także timera nad planszą

Handle\_click – pobiera pozycję kursora myszy, a także który przycisk został wciśnięty, następnie odpowiednio przesyła instrukcję do klasy Board

* 1. Board.py

Tutaj znajduje się obsługa planszy gry. Zawiera następujące metody:

Konstruktor – inicjuje planszę gry i przygotowuje ją do gry

Prepare – przygotowanie planszy do gry. Generuje obiekty klasy Field do momentu wypełnienia wymiarów planszy

Put\_bombs – Rozkłada bomby na planszy. Z pomocą random.sample() wybiera położenie bomb, następnie rozkłada je na planszy i mówi sąsiadom zainteresowanych pól, że obok jest bomba.

Fill\_neighbours – informuje sąsiadów pola, że obok niego znajduje się pole z bombą

Handle\_click – Obsługa kliknięcia. Pobiera indeks pola Field, które ma zostać odpowiednio obsłużone. Sprawdza też warunki porażki i zwycięstwa. Jeżeli odsłoni pole bez bomby oraz bez sąsiadujących bomb, to odsłania wszystkich sąsiadów tego pola

Lose – obsługa przegranej. Odsłania bomby i zaznacza źle postawione flagi.

Getlost – funkcja zwracająca informację o tym, czy gra trwa, czy została zakończona porażką

* 1. – Field.py

Obsługa pojedyńczego pola na planszy. Zawiera następujące metody:

Konstruktor – Ustawia argumenty na domyślne wartości

Clicked - Obsługa kliknięcia w pole. Nie robi nic jeżeli pole zostało kliknięte lub na polu jest flaga. Jeżeli został wciśnięty prawy przycisk myszy, to zamiast tego pole zmienia stan flagi na ekranie. W przeciwnym wypadku odsłania zawartość pola.

Flag - Obsługa stawiania i odstawiania flagi na pole.

Set\_bomb – ustawienie bomby na pole. Używane przy początkowym przygotowaniu pola do gry.

Add\_neighbouring\_bomb – ustawia informację, że na sąsiednim polu jest bomba. Informacja jest przechowywana w formie licznika, który zlicza w ilu z potencjalnie 8 sąsiadów znajduje się bomba.

Image – wybiera obraz, który ma zostać wyświetlony. Decyzja jest podejmowana na podstawie stanu pola

Image\_lost – wybiera obraz, który ma zostać wyświetlony w ramach porażki gracza.

Get\_bomb – funkcja zwracająca, czy w danym polu jest bomba