Dokumentacja gry MineSweeper

Projekt w ramach Programowania Obiektowego
Uniwersytet Wrocławski
Język C++

Michał Mikołajczyk

13 czerwca 2019

Spis treści

1 Zasady gry

Ten projekt jest klonem klasycznego *Minesweepera* znanego z wielu systemów operacyjnych Microsoftu. W tej wersji oprócz pustych pól oraz bomb, generowane są również bloki z flarami, które po naciśnięciu automatycznie odkrywają pola dookoła nich oraz oznaczą pobliskie miny flagą.

1.1 Plansza

Polem gry jest plansza zbudowana z pojedynczych bloków. Bloki mają trzy typy:

- **Blok pusty**
- **Mina**
- **Flara**

1.2 Cele i mechanika gry

- Gracz może oznaczyć dowolny blok flagą, co zapobiega odkryciu tego bloku.
- Odkrycie pustego bloku, który nie ma w pobliżu żadnych min powoduje odkrycie wszystkich jego sąsiadów.
- Jeżeli ilość sąsiadów oznaczonych flagą bloku pustego jest równa ilości pobliskich min (ukazanej na tym bloku), to możemy nacisnąć na ten blok pusty co spowoduje odkrycie wszystkich nieoznaczonych sąsiadów tego bloku. (w tym min, jeżeli bloki nie zostały oznaczone poprawnie!)
- Naciśnięcie na odkrytą flarę powoduje oznaczenie pobliskich min oraz odkrycie sąsiadujących bloków pustych.
- Flary stanowią 3% bloków na planszy.
- Miny stanowią 16% bloków na poziomie *easy* + 1% więcej na każdym kolejnym poziomie trudności – kończąc na 20% przy poziomie *custom* - *expert*.
- Gra kończy się przegraną przy odkryciu dowolnej miny.
- Gra kończy się wygraną jeżeli jedyne ukryte bloki na planszy to miny.

1.3 Sterowanie

- LMB (Lewy Przycisk Myszy) - odkrycie miny.
- RMB (Prawy Przycisk Myszy) - oznaczenie bloku flagą.
- R - szybki restart.

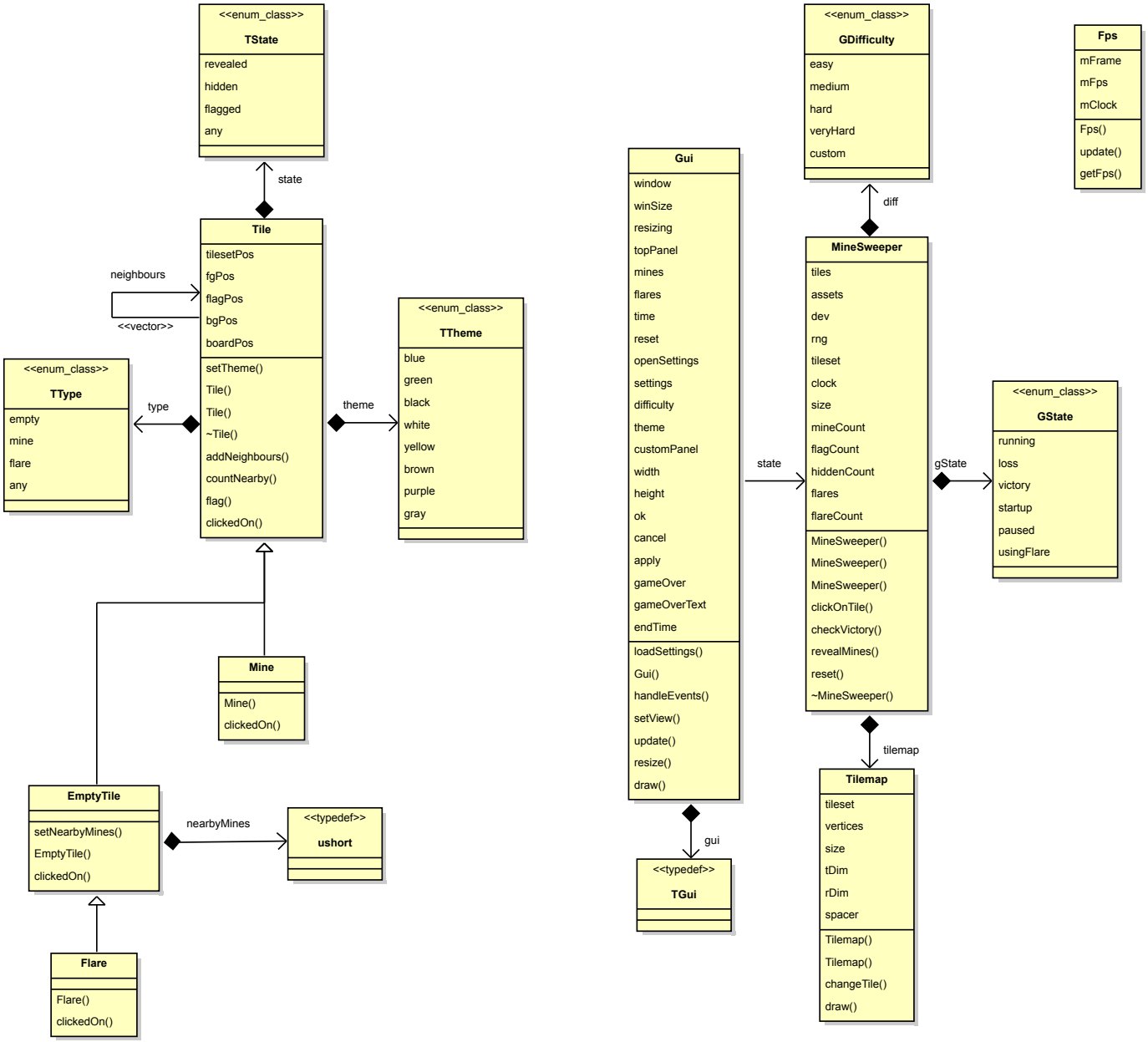
1.4 Interfejs i ustawienia

- Na samej górze okna widoczne są statystyki obecnej gry: czas, oznaczone miny oraz ilość flar dostępnych na tej planszy.
- W prawym górnym rogu znajdują się przyciski resetu gry oraz ustawień.
- Na ekranie ustawień możemy zmienić kolor bloków oraz poziom trudności.
- Po wybraniu poziomu *custom* - *expert*, w prawej stronie okna ustawień pojawią się pola, w które można wpisać własny rozmiar planszy. Maksymalny rozmiar to 100x100.

2 Podział na klasy

2.1 Diagram UML

Diagram został wygenerowany za pomocą programu Bouml.



2.2 Lista klas i ich funkcje

- **MineSweeper** - główna klasa projektu. Zawiera informacje o stanie gry oraz tekstury planszy i bloków.
- **Tilemap** - klasa dziedzicząca po klasie abstrakcyjnej sf::Drawable z biblioteki SFML. Jej zadaniem jest tworzenie i modyfikacja siatki wierzchołków, na której wyświetlane są tekstury bloków. Dzięki tej klasie gra działa dużo szybciej i jest w stanie wyświetlać dziesiątki tysięcy bloków na raz, ponieważ zamiast wywoływać funkcję draw() dla każdego bloku, możemy użyć tylko jednej, rysującej teksturę stworzoną z siatki przechowywanej w obiekcie Tilemap.
- **Tile** - klasa abstrakcyjna bloku. Zawiera funkcje pozwalające na przypisanie jej sąsiadujących z nią bloków, oznaczenie bloku flagą, zliczenie sąsiadów o danym stanie i typie oraz metodę czysto wirtualną – *clickedOn()*, która jest wywoływana gdy gracz naciska na blok. Metoda ta jest nadpisywana przez każdą klasę dziedziczącą po *Tile*.
- **Mine** - klasa dziedzicząca po *Tile* reprezentująca minę.
- **EmptyTile** - klasa dziedzicząca po *Tile* reprezentująca pusty blok. Zawiera atrybut *nearbyMines* oraz funkcję *setNearbyMines()* liczącą pobliskie miny.
- **Flare** - klasa dziedzicząca po *EmptyTile* reprezentująca flarę.
- **Gui** - klasa będąca wrapperem obiektu klasy tgui::Gui z biblioteki TGUI. Odpowiada za wczytywanie interfejsu z pliku, za podpięcie wskaźników na jego elementy oraz za jego aktualizację.
- **Klasy enumeracyjne** - te klasy odpowiadają za przypisanie wartości nazwom:
  - **TState** - stan bloku,
  - **TType** - typ bloku,
  - **GState** - stan gry,
  - **GDifficulty** - poziom trudności.
- **Fps** - mała klasa odpowiadająca za zliczanie ilości klatek na sekundę. W końcowej wersji gry nie jest ona wykorzystywana.

3 Wymagania

3.1 System operacyjny

Gra została napisana dla systemów 64 bitowych Microsoft Windows.

3.2 Zasoby

Co najmniej 100MB wolnej pamięci RAM. Podczas testowania, przy maksymalnym rozmiarze planszy (100x100), program zajmował ok. 58MB w pamięci w wersji Debug oraz ok. 42MB w wersji Release.

3.3 Biblioteki linkowane dynamicznie

- **SFML**
  Simple and Fast Multimedia Library
  https://www.sfml-dev.org/
- **TGUI_TTF**
  Texus' Graphical User Interface
  https://tgui.eu/

3.4 Środowisko programistyczne

Ten projekt został napisany przy pomocy środowiska Visual Studio 2017. Program jest kompilowany przez Visual Studio automatycznie, przy użyciu CMake.