

---

# Raport

Marcin Woźniak  
Filip Izydoreczyk  
Hubert Wrzesiński  
Przemysław Fierek

## Grupa 5

11 czerwca 2019

Repozytorium: [https://github.com/linux923344/autonomiczny\\_saper/](https://github.com/linux923344/autonomiczny_saper/)

Spis klas użytych w projekcie:

### 1. Board

- (i) Board - jako argumenty przyjmuje X i Y, które są rozmiarem renderowanego okna. Klasa odpowiada za renderowanie mapy i zarządzanie obiektami na niej.
- (ii) Direction - enum, który daje nam informacje, w którym kierunku ma się poruszać postać.
- (iii) DirectionCalculator - obliczanie następnego punktu na podstawie podanych wcześniej x i y.
- (iv) EquipmentGui - klasa odpowiedzialna za grafikę Ekwipunku
- (v) EquipmentGuiControl - klasa odpowiedzialna za zbieranie narzędzi i umieszczanie ich w ekwipunku.
- (vi) GameStarter - odpowiada za wystartowanie całego programu, po wybraniu odpowiednich opcji z menu.
- (vii) MapReader - argumentem przy jej tworzeniu jest plansza czyli obiekt typu Board, na której MapReader będzie tworzył obiekty, które są przechowywane w zewnętrznym pliku.
- (viii) Point - klasa punktu
- (ix) WalkingType - typ chodzenia (Za pomocą uczenia maszynowego, lub algorytmów chodzenia)

### 2. MapObjects

## (i) Bombs:

- BombRed
  - BombBlue
  - BombYellow
- } Obiekty odpowiedzialne za renderowanie oraz logikę bomby

(ii) Saper - przechowuje informacje potrzebne do wyrenderowania Saper, podnoszenie rzeczy do ekwipunku oraz jakie zadania musi wykonać.

(iii) Stone - klasa, która jest odpowiedzialna za informacje potrzebne do wyrenderowania kamienia.

(iv) Tool - klasa, która jest odpowiedzialna za informacje potrzebne do wyrenderowania narzędzia.

(v) Water - klasa, która jest odpowiedzialna za informacje potrzebne do wyrenderowania wody.

### 3. MenuWindow

(i) MenuWindow - klasa odpowiedzialna za wygenerowanie menu, dzięki któremu możemy wybrać mapę, po której chcemy się poruszać, oraz typ algorytmu, za pomocą którego chcemy przejść daną mapę.

(ii) AlgorithmType - enum, z pomocą którego wybieramy algorytmy w MenuWindow.

### 4. Path Finder

(i) BestFirstSearch

- Vertex - tworzy wierzchołki grafu
- GraphCreatorBestFS - tworzy graf BestFirstSearch
- GraphBFS - przejście po grafie BestFirstSearch

(ii) BreathFirstSearch

- Vertex - tworzy wierzchołki grafu
- GraphCreatorBFS - tworzy graf BFS
- GraphBFS - przejście po grafie BFS

(iii) DepthFirstSearch

- Vertex - tworzy wierzchołki grafu
- GraphCreator - tworzy graf DFS
- Graph - przejście po grafie DFS

### 5. Pathfinder- wywołuje poszczególne algorytmy na planszy

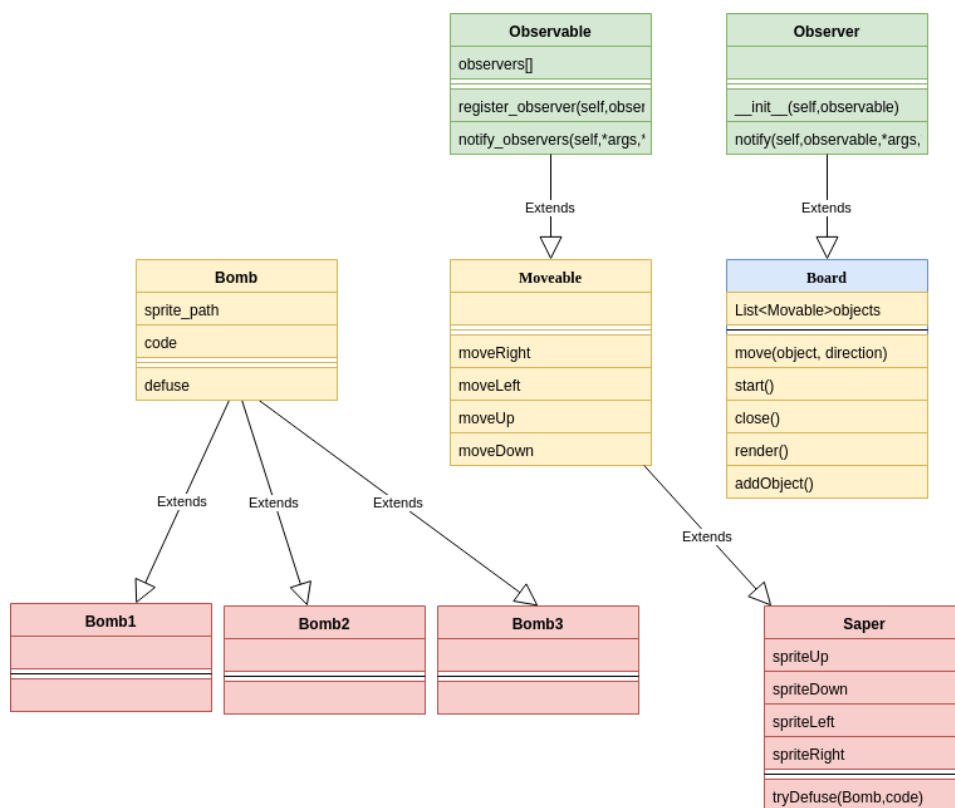
### 6. VowpalVabbit

(i) DataCreator - klasa która tworzy dane uczące dla Vowpal Wabbit

- (ii) CellState - stan komórki
- (iii) VowpalPredicter - przewidywanie kolejnych kroków za pomocą Vowpala

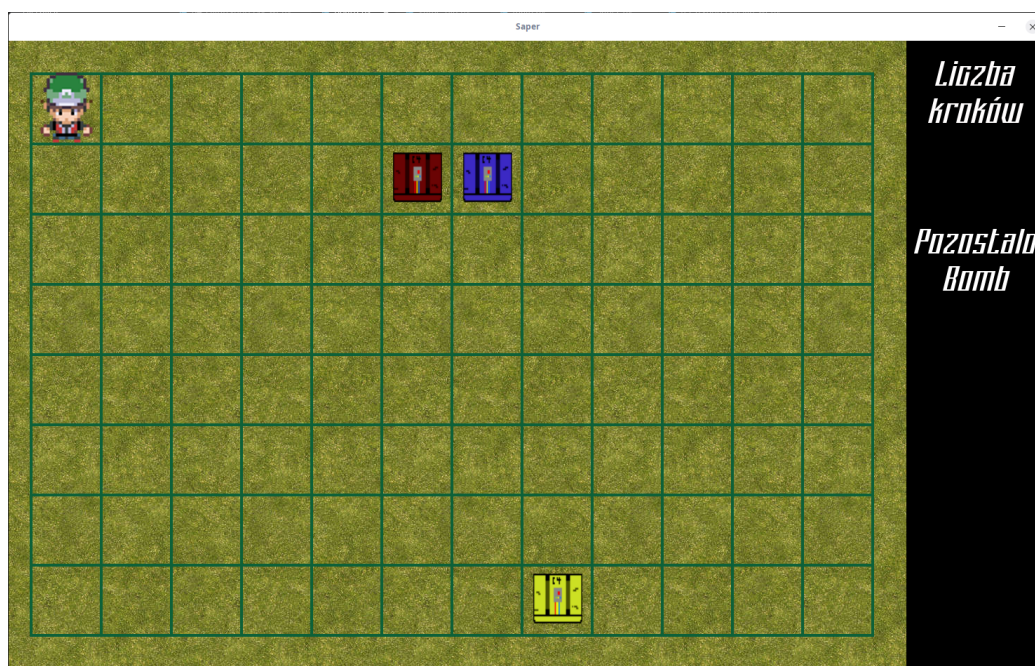
27.03.2019 r.

# 1. Zaprojektowanie diagramu klas.



2. Na podstawie diagramu, została stworzona klasa główna oraz jej podklasy.

3. Został stworzony szablon planszy razem z umieszczonymi na niej bombami.



Na załączonym zrzucie ekranu widoczny jest saper czyli nasz Agent, oraz trzy kolorowe obiekty (bomby), które nasz agent ma za zadanie je rozbroić.

**10.04.2019 r.**

W tym dniu została dokończona reprezentacja wiedzy w naszym projekcie. Przedstawimy ją teraz:

1. Posiadamy trzy rodzaje bomb (czerwona, żółta, niebieska).
2. Aby rozbroić bombę czerwoną musimy posiadać narzędzie dodatkowo zmieścić się w wyznaczonym czasie. Jednostka czasu w naszym świecie jest jeden krok.
3. Aby rozbroić pozostałe bomby Saper musi tylko do nich podejść, oczywiście jak najkrótszym czasie.
4. Saper będzie wyznaczał drogi za pomocą algorytmu przeszukiwania grafu.

**15.04.2019 r.**

W tym dniu został zaimplementowany wyznaczanie ścieżki za pomocą algorytmu przeszukiwania DFS oraz kilka klas.

**17.04.2019 r.**

Spisanie raportu i wypisanie wszystkich klas znajdujących się w projekcie.

**30.04.2019 r.**

Dodanie ekwipunku i ustalenie zasad działania bomb. Bomby będą miały określony czas, a narzędzia będą potrzebne do rozbicia ich.

**6.05.2019 r.**

Implementacja algorytmu BFS.

**14.05.2019 r.**

Implementacja algorytmu Best-First Search.

**28.05.2019 r.**

Dodanie klasy tworzącej dane uczące. Agent przechodzi graf, i z każdym krokiem zapisuje stan pól znajdujących się dookoła niego w kwadracie 7x7, do pliku.

**29.05.2019 r.**

Ulepszenie algorytmu Best-First Search. Teraz nasz agent przechodzi graf zbierając po drodze wszystkie narzędzia znajdujące się na mapie

**01.06.2019 r.**

Wykorzystanie zapisanych danych uczących przy Wopal Wabbicie, a także dodanie licznika do bomb.

**02.06.2019 r.**

Stworzenie większej ilości map do uczenia, a co za tym idzie, stworzenie większej ilości danych uczących.

**03.06.2019 r.**

Agent jest nauczony, ale nie działa perfekcyjnie.

**Opis zastosowania drzew decyzyjnych**

**Opis zastosowania Volpal Wabbit**