Sprawozdanie z Projektu
Tanks

Daniel Ślusarczyk i Jakub Łaba

09.06.2021

Streszczenie

Dokument stanowi sprawozdanie z projektu *"Tanks"*, z wyszczególnionym opisem aktualnego stanu projektu, zmian względem specyfikacji oraz wnioskami.

Spis treści

1 Zarys Projektu

Projekt *"Tanks"* zakładał stworzenie programu dedykowanego dwóm graczom rozmieszczonym po dwóch stronach okna rozgrywki, których zadaniem jest strzelanie do komórek w środkowej części okna za pomocą przypisanych czołgów. Gracze maja możliwość wertykalnego poruszania się swoimi jednostkami, oraz obracania lufą czołgu. Każdy pojazd strzela pociskami o cechach możliwych do określenie przez plik konfiguracyjny.
Kwadratowe komórki, które są obiektem przyznawania punktów graczom, dzielą się na trzy rodzaje:

- **Zwykła komórka** – kwadratowy obiekt o wszystkich krawędziach wrażliwych na kontakt z pociskiem. Punkty za jej unicestwienie przyznawane są graczowi zadającemu końcowe obrażenia.
- **Bomba** – obiekt posiadający niewrażliwe krawędzie z pominięciem górnej, która jest jedynym sposobem na zastrzelenie tej komórki. Dokonanie tego kończy grę.
- **Kolonia** – zbiór maksymalnie pięciu zwykłych komórek w losowej konfiguracji. Gracz otrzymuje zsumowane punkty wszystkich komórek wchodzących w skład kolonii po zastrzeleniu ostatniej komórki w zbiorze.

Wszystkie wyżej wymienione komórki poruszają wertykalnie w dół w ramach okna rozgrywki. Podobnie jak pociski posiadają parametry modyfikowalne z poziomu pliku konfiguracyjnego, a ich najważniejszą cechą jest wartość potrzebna do zastrzelenia znajdująca się w centralnej części komórki.
Gra w sposób regularny zwiększa trudność rozgrywki poprzez zwiększanie wartości potrzebnej do unicestwienia komórki, szybkości komórek i wystrzeliwanych pocisków, oraz zmniejszanie promienia pocisku i długości boku komórki. Koniec gry przewidują dwa scenariusze: upłyniecie określonego czasu rozgrywki lub zastrzelenie komórki bomby. Grę wygrywa gracz o większej ilości punktów w momencie zakończenia.

2 Struktura programu

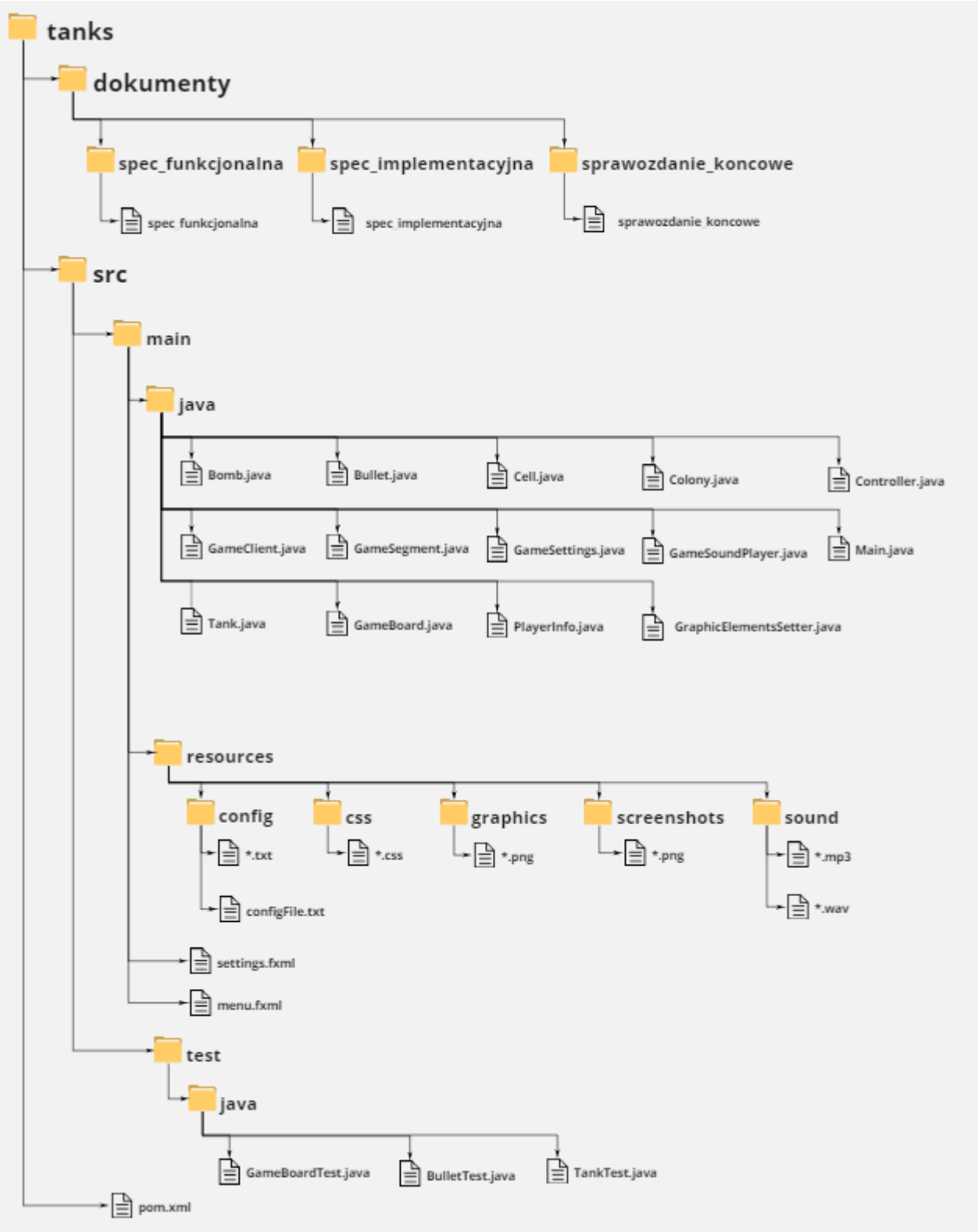
2.1 Struktura katalogów

Folder *tanks* jest głównym folderem projektowym, w ramach którego mieszczą się wszystkie pliki związane z projektem. Zawiera dwa foldery i jeden plik:

- Dokumenty
  Folder *dokumenty* przechowuje specyfikację funkcjonalną i implementacyjną, oraz sprawozdanie końcowe. Każdy z tych dokumentów znajduje się w osobnych folderze, który zawiera pliki związane wyłącznie z danym dokumentem.

- Src
  Folder *src* jest katalogiem związanym z kodem i działaniem przygotowanego oprogramowania. W podkatalogu o ścieżce *test/java* przechowuje przygotowane testy automatyczne oprogramowania. Natomiast w podkatalogu *main* przechowywany jest kod źródłowy projektu, folder do zapisu wykonanych zrzutów ekranu i pliki niezbędne do działania programu, a w szczególności pliki konfiguracyjne.

- Pom.xml
  Plik zawierający informacje o projekcie i szczegóły używane przez narzędzie automatyzujące budowę oprogramowania *Maven*.



Rysunek 1: Schemat struktury katalogów

3 Szczegóły oprogramowania

3.1 Narzędzie automatyzujące budowę "Maven"

Do stworzenia gry został wykorzystany *Maven* – jest to narzędzie upraszczające pracę nad tworzeniem aplikacji, zwłaszcza korzystających z dużej ilości modułów – co w przypadku mocno podzielonego na moduły frameworku graficznego *JavaFX* jest pomocne. Najważniejsze moduły załączone w aplikacji za pomocą *Maven*-a to:

- **JavaFX Graphics**
- **JavaFX Controls**
- **JavaFX FXML**
- **JavaFX Media**
- **JUnit5**

3.2 Automatyczne testy

Aplikacja została przetestowana za pomocą testów jednostkowych z wykorzystaniem biblioteki *JUnit* w wersji `5.7.0`. Przetestowane zostały metody odpowiadające za poruszanie oraz usuwanie poszczególnych komponentów rozgrywki.

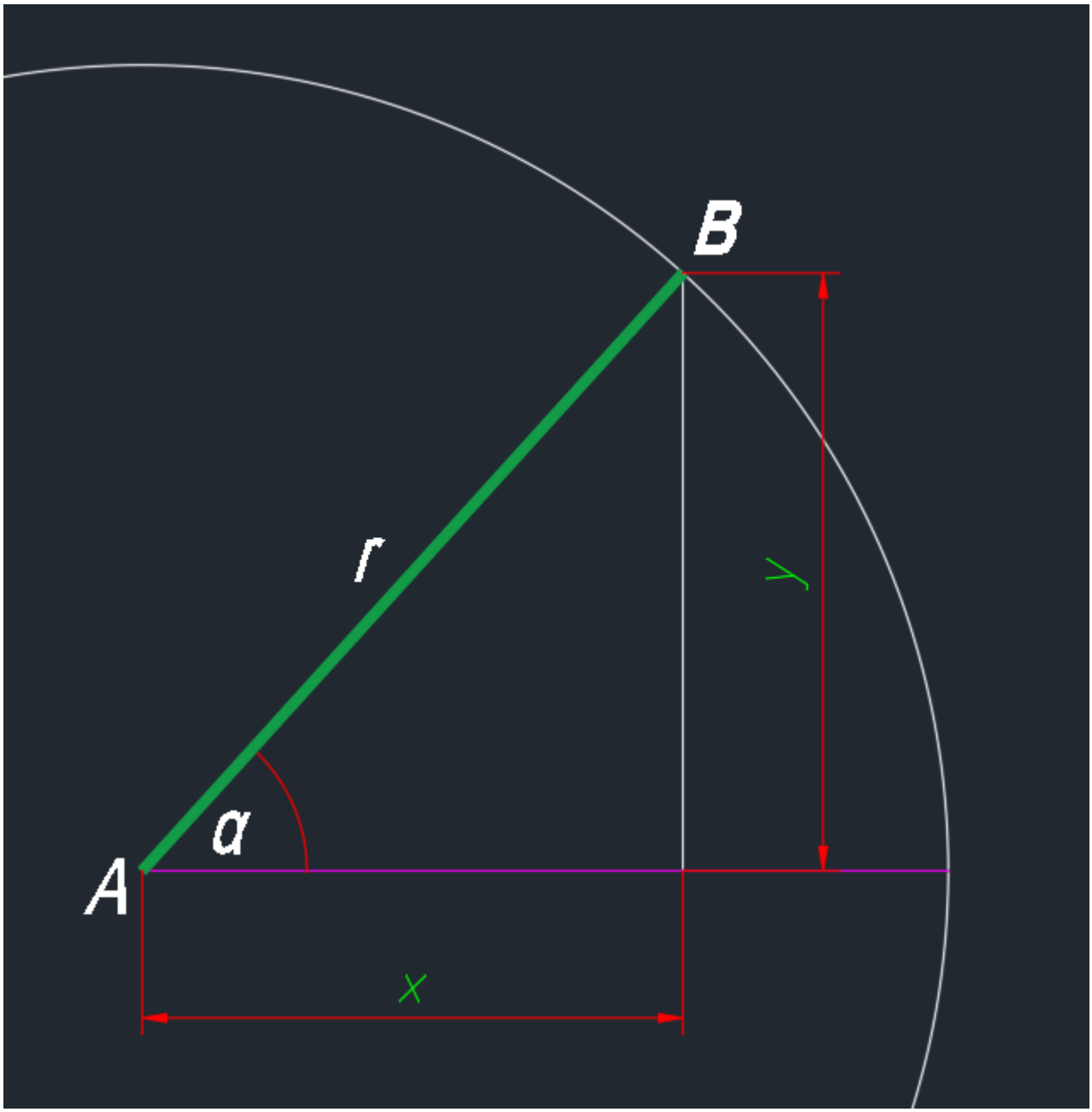
3.3 Zasady działania elementów

Kolonie

Generowanie kolonii odbywa się na podstawie kwadratu o wymiarach 3x3 komórki. Środkowe pole kwadratu jest zawsze zajmowane przez komórkę, a dodatkowo jest losowane od 1 do 4 komórek, które zajmują losowe miejsca dookoła niej. Wszystkie komórki zmniejszają się o wartość DH1 co T1 sekund. Unicestwienie kolonii wymaga unicestwienia wszystkich komórek należących do zbioru. Komórki, które zostały trafione wystarczającą ilość razy i należą do kolonii są wyświetlane z cyfrą 0 i stają się niewrażliwie na pociski. Trafienie ostatniej komórki powoduje zniknięcie wszystkich elementów kolonii i naliczenie punktów graczowi, który tego dokonał.

Pociski

Bullet jest klasą dziedziczącą po *GameSegment* – został on rozszerzony o dane dotyczące wektora, opisującego tor ruchu danego pocisku. Wektor jest obliczany już w konstruktorze, który jest wywoływany w metodzie *shoot()* klasy *Tank* – pocisk powstaje w momencie jego wystrzelenia. Poniższy schemat wizualizuje kilka istotnych zależności, wykorzystanych w celu odpowiedniego umieszczania i poruszania się pocisków. Punkt A jest jest współrzędnymi czołgu przechowywanymi w polach klasy *Tank*, natomiast punkt B reprezentuje czubek lufy – miejsce w którym w momencie wystrzału powinien pojawić się pocisk. Długość lufy jest reprezentowana przez r, ta wartośc również jest przechowywana jako pole w klasie *Tank*. Wektorem ruchu pocisku jest wektor $\vec{AB}$.



Rysunek 2: Schemat przedstawiający lufę czołgu (zaznaczona kolorem zielonym)

Przyjmując

$$A = (x_a, y_a)$$
$$B = (x_b, y_b)$$

Otrzymujemy

$$x = x_b - x_a$$
$$y = y_a - y_b$$

$$\vec{AB} = [x_b - x_a, y_b - y_a] = [x, -y]$$

Wówczas

$$sin\alpha = \frac{y_a - y_b}{r}$$

$$cos\alpha = \frac{x_b - x_a}{r}$$

Zatem

$$x = rcos\alpha$$

$$y = rsin\alpha$$

$$\vec{AB} = [rcos\alpha, -rsin\alpha]$$

$$B = (x_a + x, y_a - y)$$

3.4 Licencje elementów

Przygotowane oprogramowanie korzysta z plików graficznych i dźwiękowych udostępnionych przez autorów na zasadach licencji pozwalających na legalne wykorzystanie ich w projekcie *"Tanks"*.

Informacje o poszczególnych elementach:

- Wszystkie elementy graficzne:
  Źródło: *https://opengameart.org/content/tank-2*
  Autor: KASTLE Knight
  Licencja: CC 1.0

- Muzyka w tle:
  Źródło: *https://freesound.org/people/bipwave/sounds/505985/*
  Autor: bipwave
  Licencja: CC 1.0

- Muzyka końca gry:
  Źródło: *https://freesound.org/people/kimp10/sounds/341578/*
  Autor: kimp10
  Licencja: CC 1.0

- Efekty dźwiękowe:
  Źródło: *https://mixkit.co/free-sound-effects/music/*
  Licencja: Mixkit Sound Effects Free License

4 Przegląd programu

Przedstawione poniżej elementy prezentują różne funkcjonalności oprogramowania.

4.1 Elementy obsługi gry

Okno menu

Poniższy rysunek przedstawia okno menu widziane jako pierwszy element po uruchomieniu oprogramowania.

Rysunek 3: Okno menu

Okno ustawień

- Pierwsza zakładka
  Z poziomu pierwszej zakładki użytkownik ma możliwość ustawienia niektórych elementów działania program – dostosowanie głośności dźwięku, wczytanie własnego pliku konfiguracyjnego i zaznaczenie wykonywania zdjęcia po zakończonej rozgrywce.

Rysunek 4: Zakładka Game Properties

- Druga zakładka
  Z poziomu drugiej zakładki użytkownik ma możliwość dostosowania poszczególnych elementów pliku konfiguracyjnego. Dodatkowo na samym dole znajduje się przycisk, który po podaniu żądanej nazwy pliku jest w stanie stworzyć plik konfiguracyjny o podanej nazwie w formacie zgodnym z oprogramowaniem i zapisać zmiany.

Rysunek 5: Zakładka Manual Configuration

Podanie wartości o niezgodnym formacie i próba zapisu do pliku powoduje podświetlenie niewłaściwych pól na czerwono i zaakceptowanie tylko tych prawidłowych.

Rysunek 6: Zakładka Manual Configuration

- Trzecia zakładka
  Z poziomu trzeciej zakładki użytkownik ma możliwośc przypisania w sposób dowolny sterowania gry dla obu użytkowników.

Rysunek 7: Zakładka Manual Configuration

Próba przypisania sterowania do wykorzystywanego przycisku powoduje podświetlnie na czerwono miejsca, w którym jest przypisany i niezapisanie zmian.

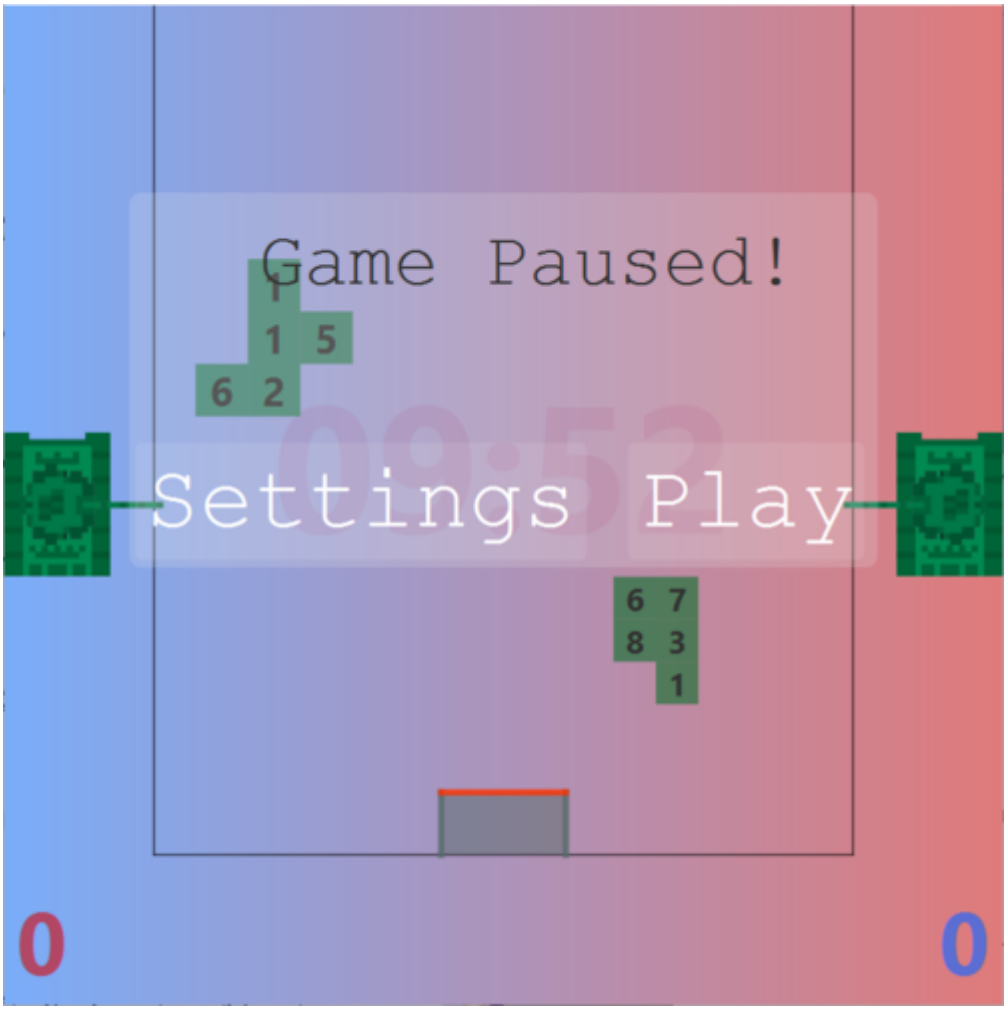
Rysunek 8: Zakładka Manual Configuration

Elementy Statyczne Rozgrywki

- Koniec gry
  Zakończenie rozgrywki prezentowane jest przez poniższe okno. Udostępnia ono informacje w formie diagramu kołowego o ilości punktów uzyskanych przez poszczególnych graczy i daje możliwość wyjścia z gry, lub ponownego uruchomienia rozgrywki. Jeśli w ustawieniach zostało zaznaczone miejsce ustawiające wykonanie zrzutu ekranu to w folderze *'resources/screenshots'* zostanie zapisy zrzut ekranu tego okna.

Rysunek 9: Okno końca gry

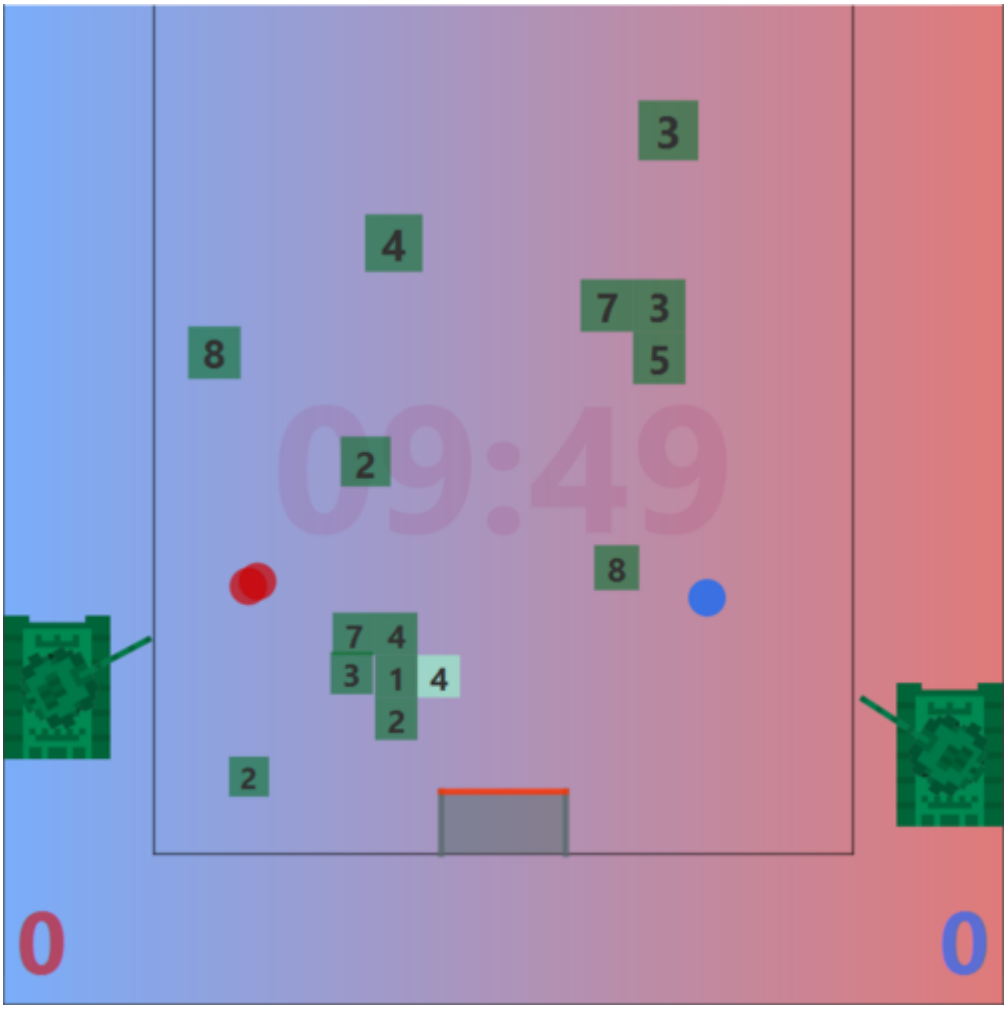
- Wstrzymanie gry
  Naciśnięcie przycisku "P" powoduje zatrzymanie działania gry i oczekiwanie na dalsze czynności użytkownika. Z poziomu wyświetlanego okna można wznowić grę, lub przejść do okna ustawień.



Rysunek 10: Okno Wstrzymania Gry

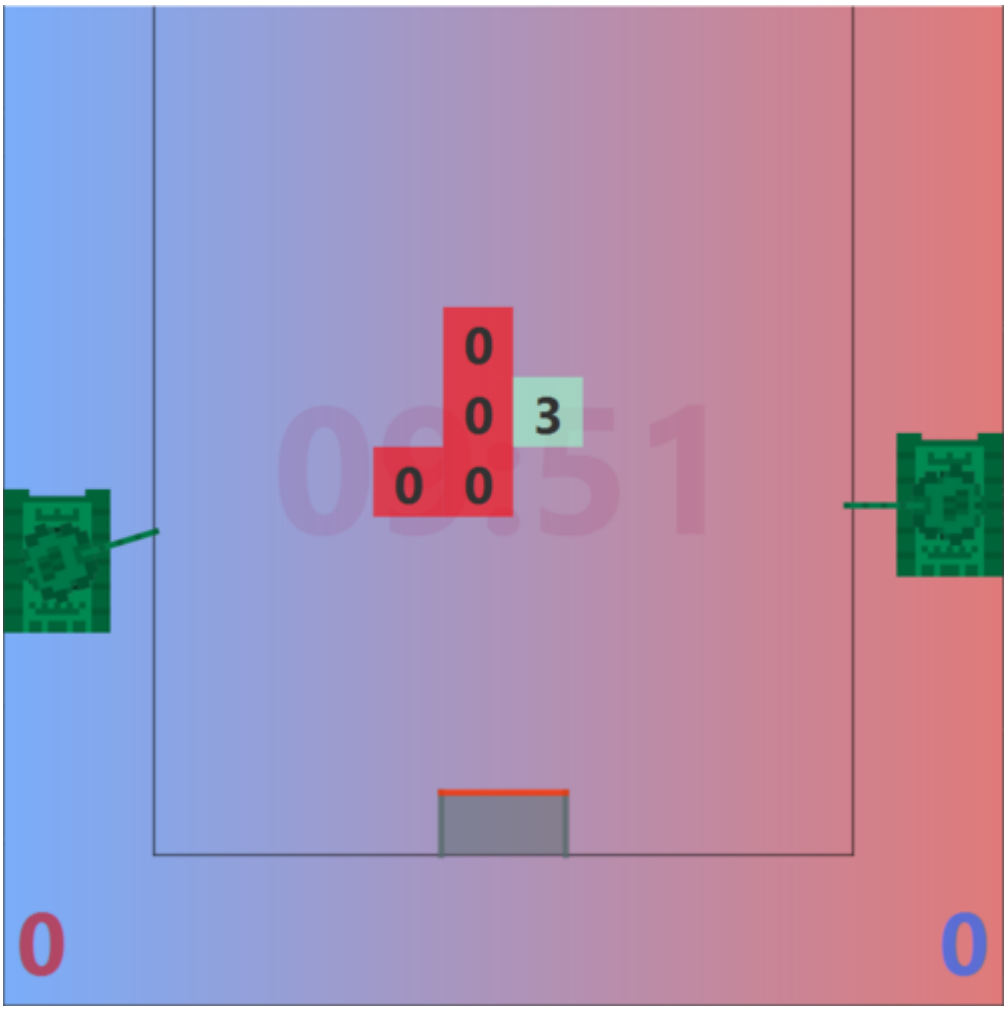
4.2 Rozgrywka

- Okno wyświetlane podczas stanardowej rozgrywki zawiera takie elementy jak: punktacje obu graczy, czołgi obu graczy, czas do końca rozgrywki, komórki, kolonie i miejsce wyświetlania komunikatów o ewentualnych błędach.



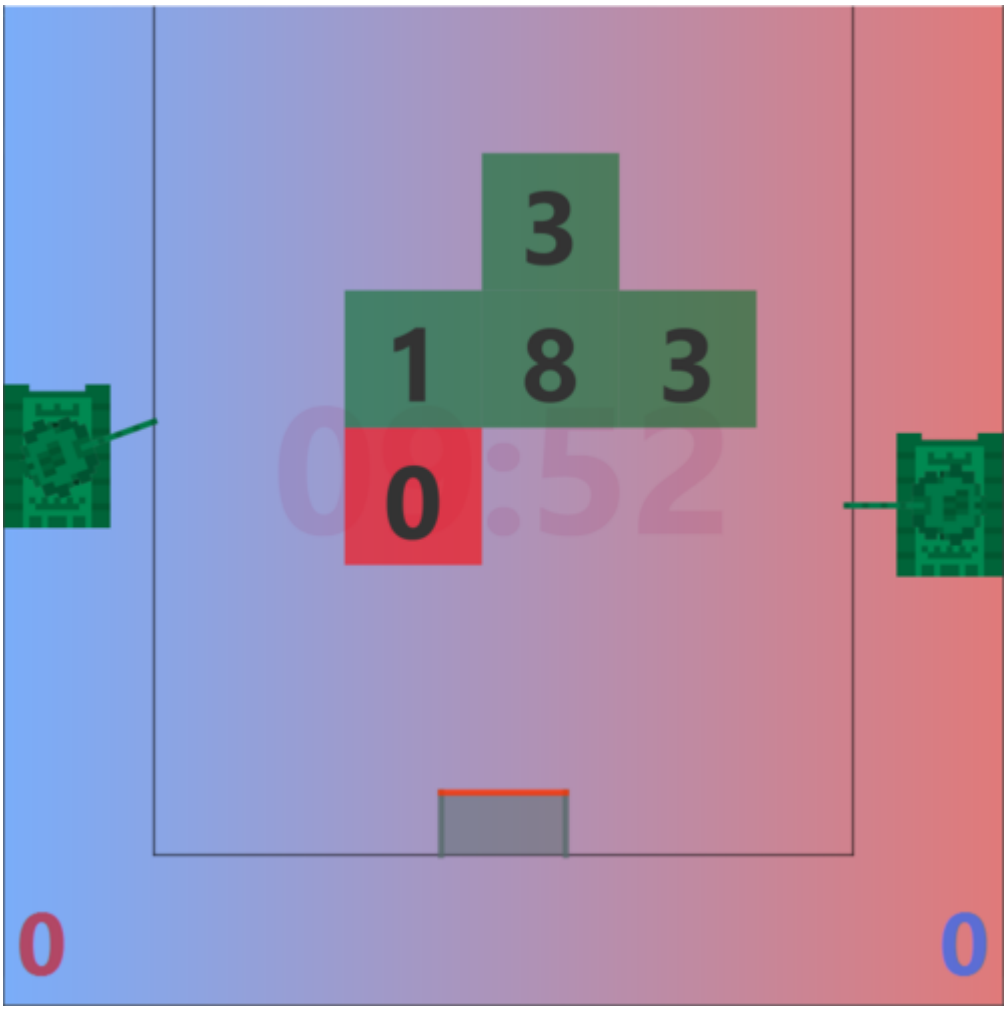
Rysunek 11: Przykład rozgrywki

- Kolonia z ostatnią żywą komórką wyświetlana jest w następujący sposób.



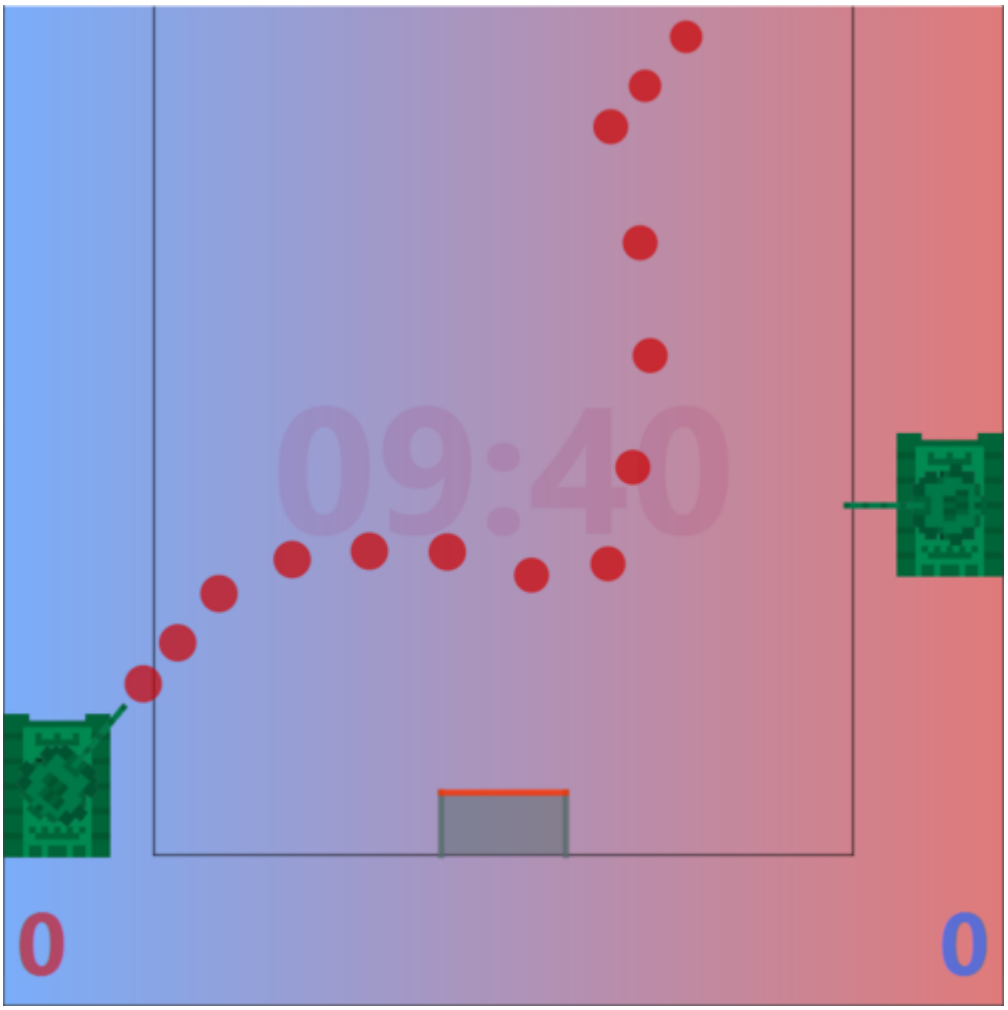
Rysunek 12: Przykład rozgrywki

- Kololonia wyświetlana dla wartości H1 równej 120.



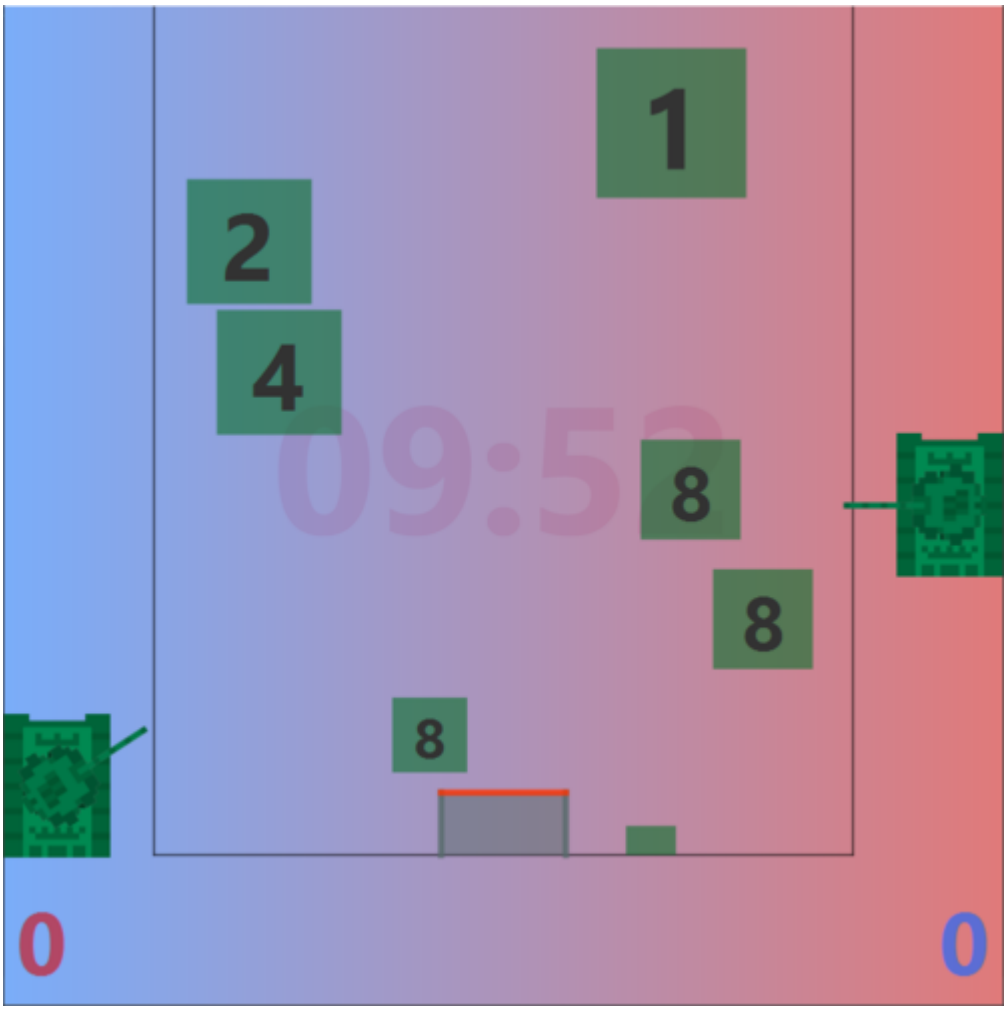
Rysunek 13: Przykład rozgrywki

- Pociski gracza dla wartości X1 równej 1000 i V1 równej 8.



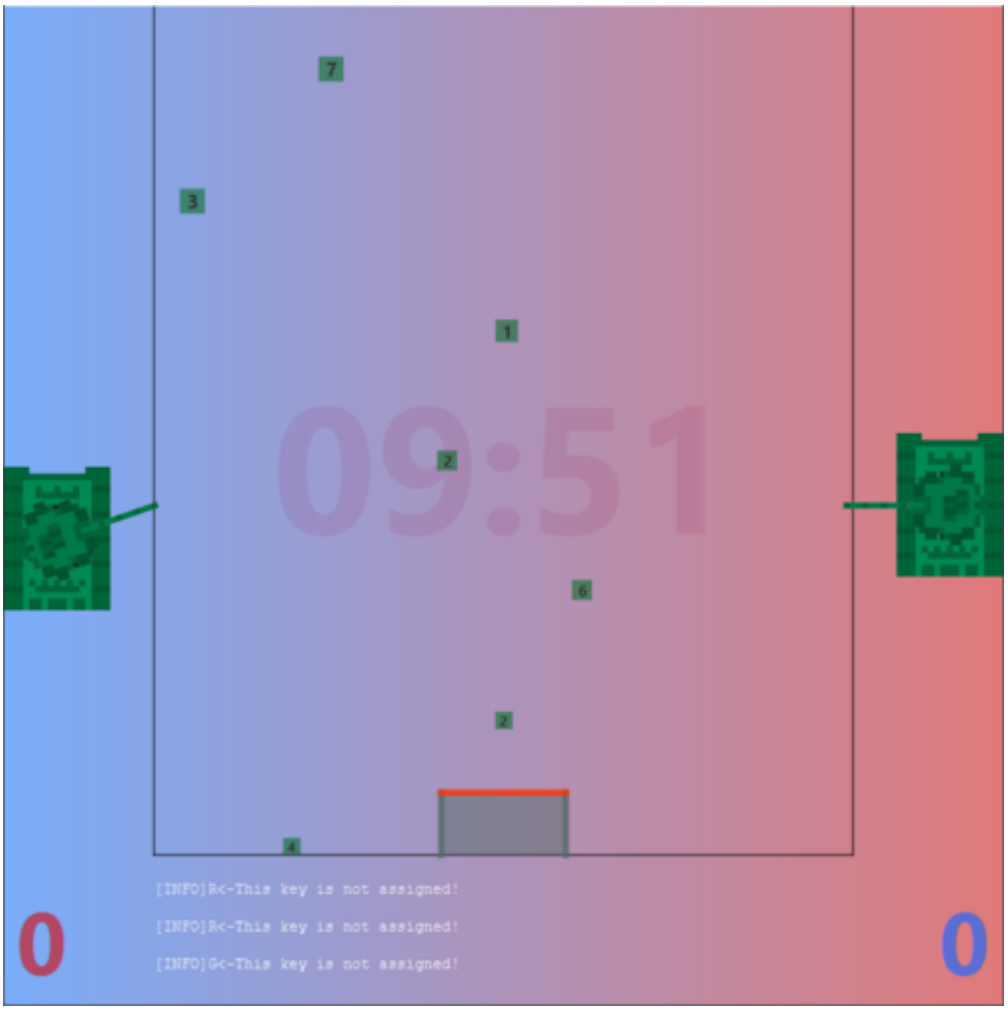
Rysunek 14: Przykład rozgrywki

- Pojedyncze komórki dla wartości H1 równej 120, DH1 równej 20 i V2 równej 100.



Rysunek 15: Przykład rozgrywki

- Okno rozgrywki z komunikatami o naciśnięciu przycisków, które nie są przypisane do żadnego sterowania.



Rysunek 16: Przykład rozgrywki

5 Zmiany względem specyfikacji

Założeniem projektu od samego początku powstawania było zachowanie ścisłości między pisanym programem, a specyfikacją funkcjonalna i implementacyjną. Proces powstawania elementów projektu zweryfikował dokładnie wszystkie założone idee i spowodował zmianę niektórych elementów.

5.1 Plik konfiguracyjny

Plik konfiguracyjny został wzbogacony o dwie dodatkowe zmienne:

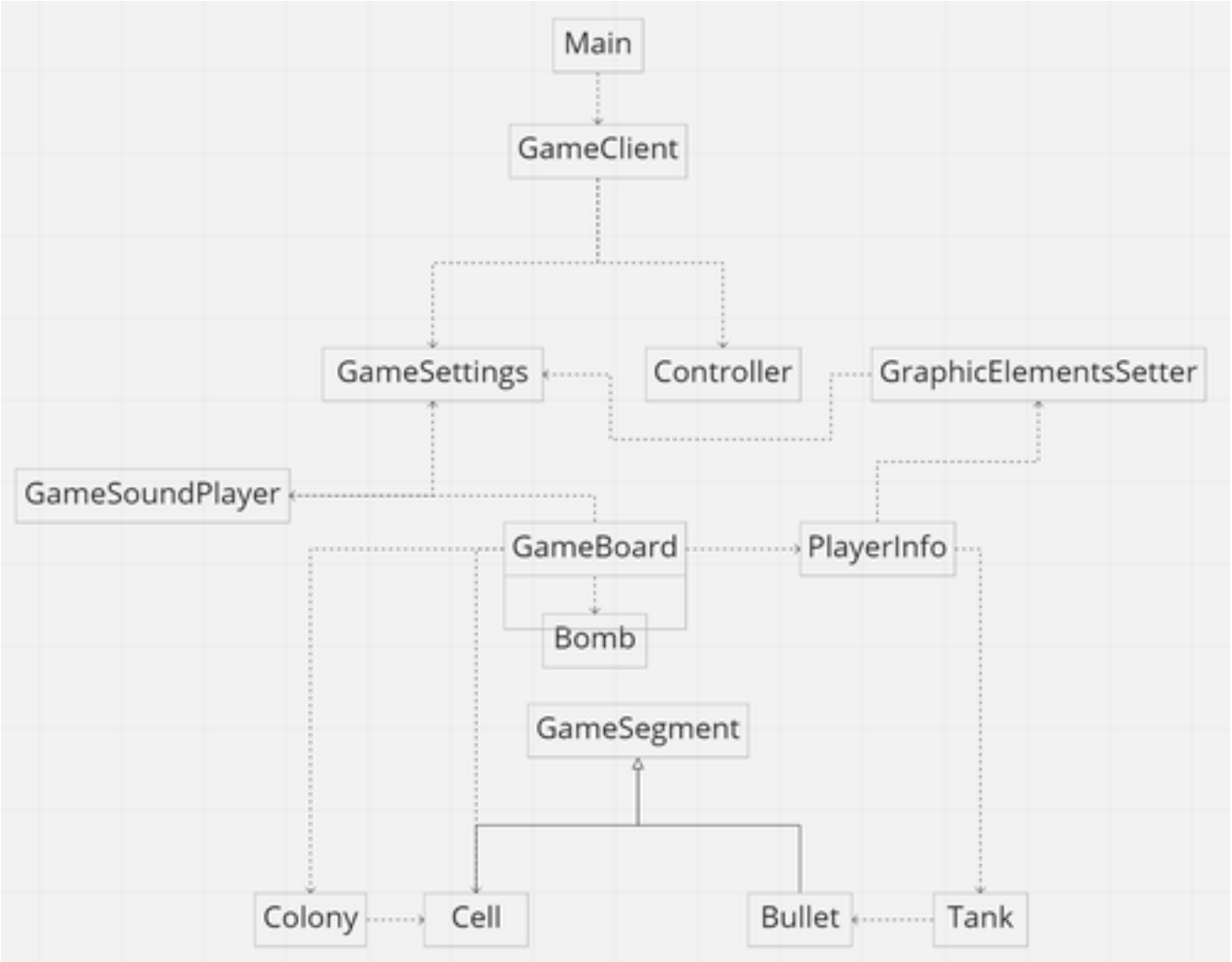
- [T4] TimeBetweenCellGenerating [liczba rzeczywista] – pozwala określić czas pomiędzy kolejnymi wygenerowanymi komórkami.
- [T5] TimeBetweenColonyGeneration [liczba rzeczywista] – pozwala określić czas pomiędzy kolejnymi wygenerowanymi koloniami.

5.2 Funkcjonalności programu

W funkcjonalnościach programu zrezygnowano z trzech predefiniowanych trybów trudności. Niemniej jednak, zostało dododane znacznie bardziej rozbudowane okno ustawień. Dodatkowo okno błędów ma możliwość nie tylko informowania o ewentualnych błędach, ale również np. o naciśnietych przyciskach, które nie są nigdzie przypisane.

5.3 Aktualny uproszczony diagram klas

W trakcie prac nad projektem, zostały utworzone dwie nowe klasy – *GraphicElementsSetter* oraz *GameSoundPlayer*. *GraphicElementsSetter* jest odpowiedzialny za tworzenie graficznych komponentów gry – klasa ta powstała w celu zredukowania ilości metod w klasie *Controller*. Dodanie do gry dźwięku wymusiło stworzenie klasy *GameSoundPlayer*, obsługującej metody związane z odtwarzaniem go.



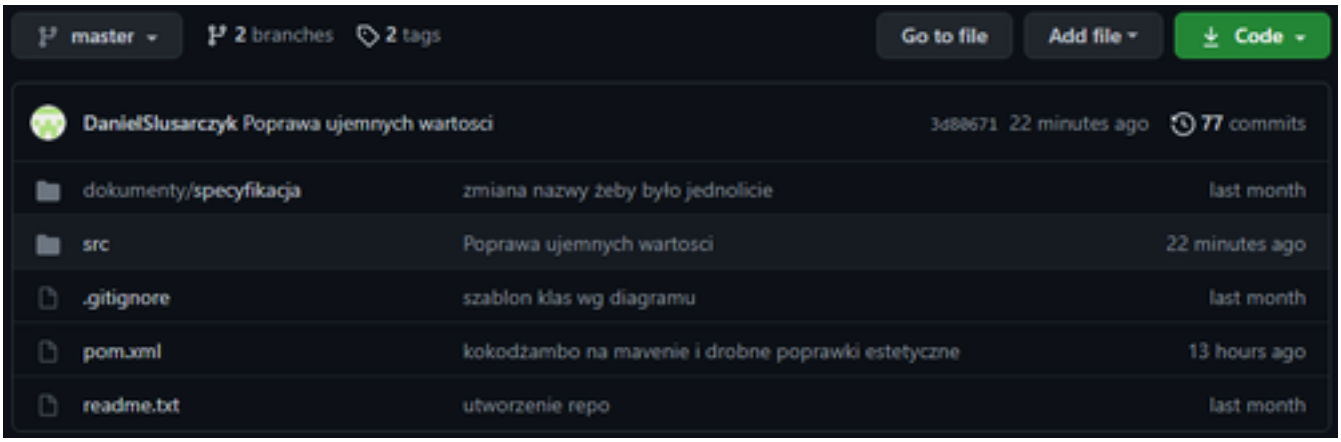
Rysunek 17: Uproszczony diagram UML, pokazujący zależności pomiędzy klasami

6 Podsumowanie współpracy

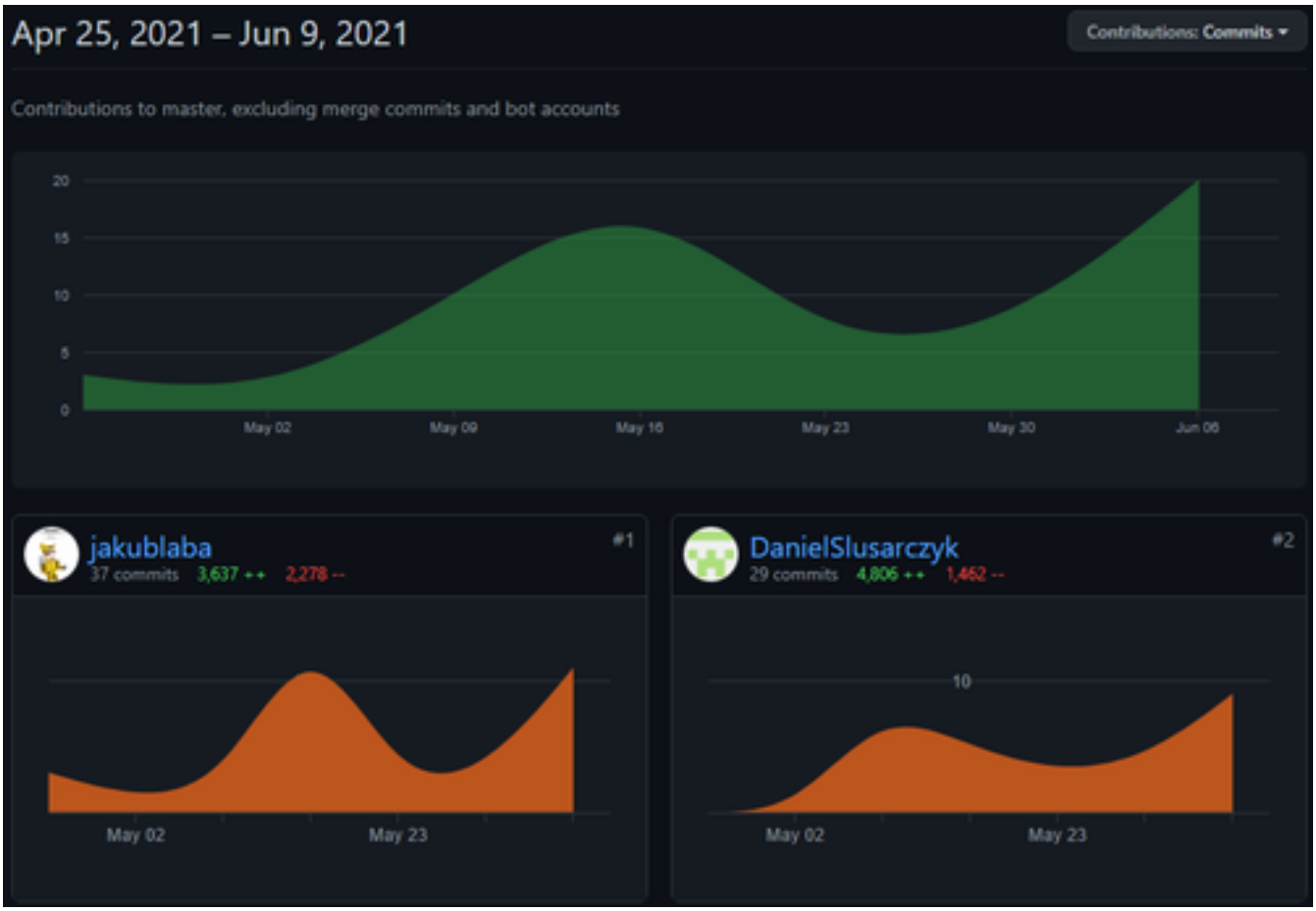
Współpraca w obrębie projektu *Tanks* przebiegała bez zauważalnych problemów. Głównymi czynnikiami wyraźnie ułatwiającymi współpracę było korzystanie z podobnych narzędzi programistycznych i doświadczenie wyniesione z poprzednich projektów.

6.1 Systematyczność

Równolegle z uczelnianym repozytorium *Projektor* korzystaliśmy również z *Githuba* głównie z powodu możliwości dodania wielu kluczy `SSH` dla pojedynczego użytkownika – istotność potrzeby takiej funkcjonalności wynika z faktu, że obu autorów kodu pracowało na wielu urządzeniach bądź maszynach wirtualnych.
Dane dotyczące repozytorium `https://github.com/jakublaba/tanks` na dzień `9.06.2021`:

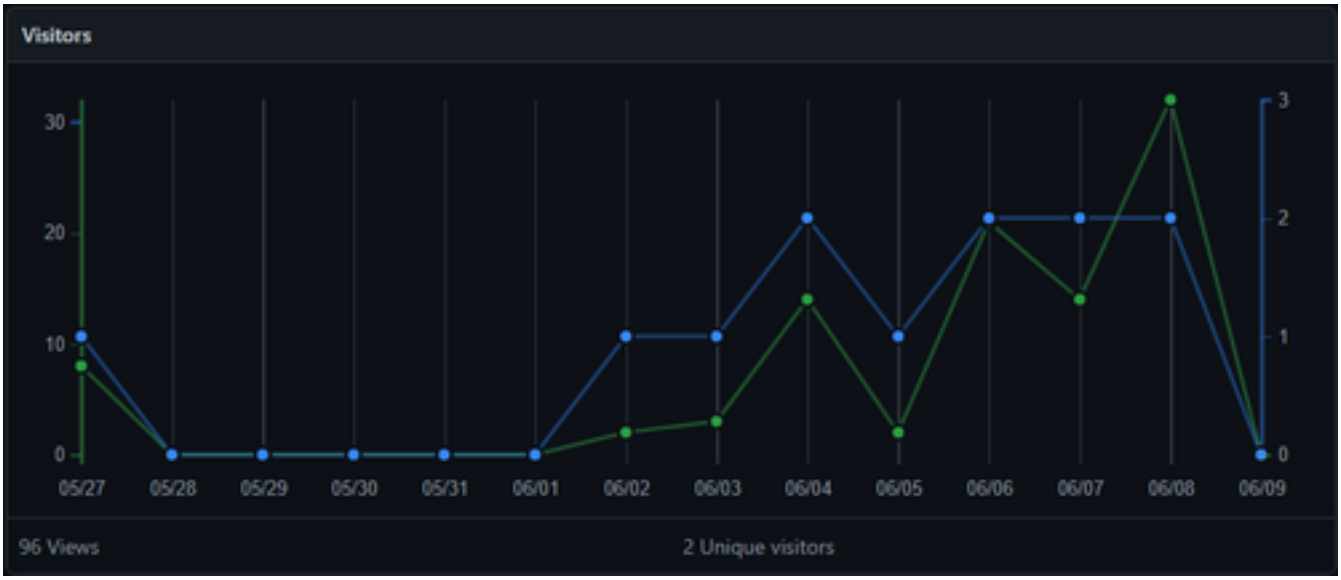


Rysunek 18: Strona główna repozytorium, ukazująca ilość commit-ów

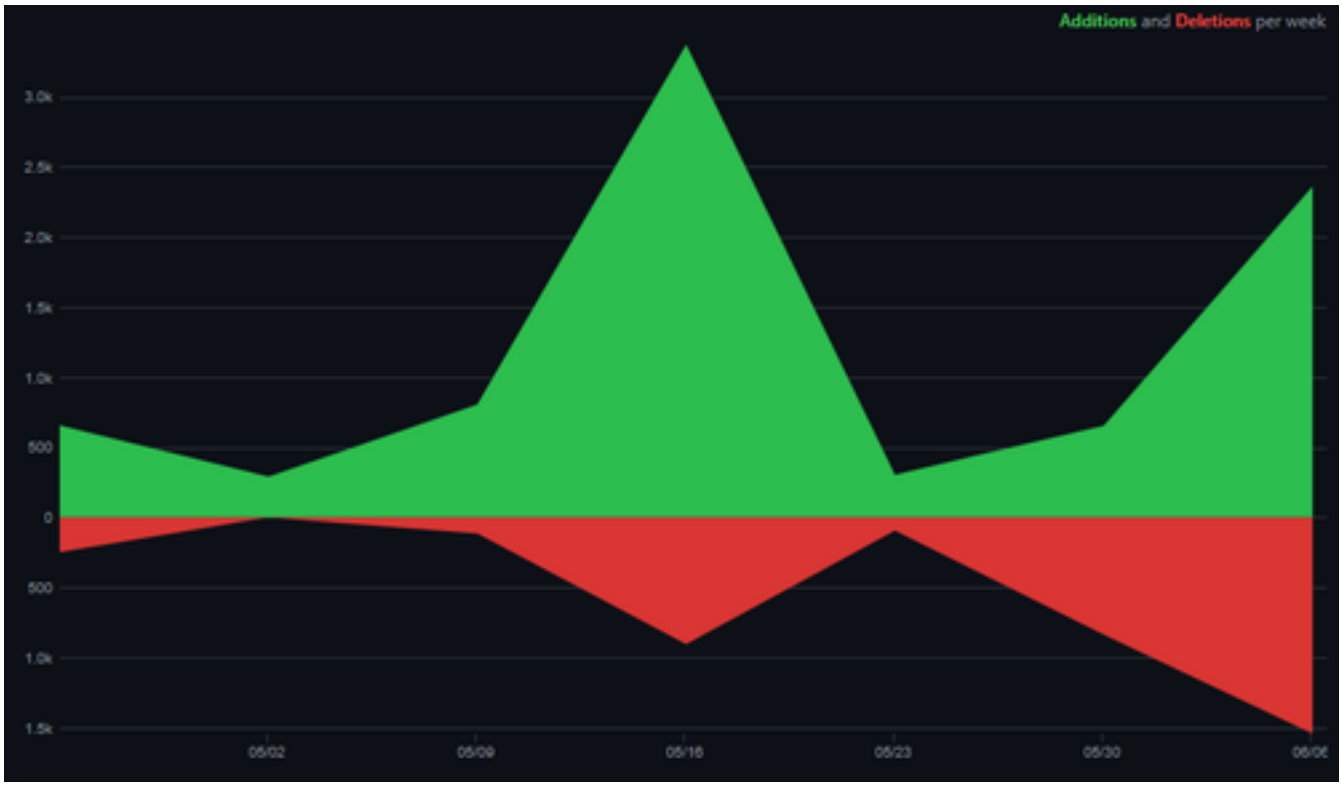


Rysunek 19: Wykresy ukazujące częstotliwośc commit-ów poszczególnych użytkowników repozytorium

Rysunek 20: Wykresy ukazujące ogólną częstotliwość commit-ów



Rysunek 21: Wykres ukazujący częstotliwość odwiedzania repozytorium



Rysunek 22: Wykres ukazujący średnie tygodniowe ilości edytowanych linii kodu

7 Podsumowanie projektu

Projekt "*Tanks*" został w całości przygotowany od 15.04.2021 do 09.06.2021. W ramach niego powstała działająca gra dedykowana dwóm graczom, specyfikacja implementacyjna, specyfikacja funkcjonalna oraz sprawozdanie końcowe. Przygotowane rozwiązanie korzysta zarówno z elementów graficznych, jak również wielu narzędzi programistycznych dostarczanych przez język programowania obiektowego „Java". Poprzez modyfikowalny plik konfiguracyjny każdy odbiorca oprogramowania ma możliwość w prosty sposób dostosować wiele elementów rozgrywki zgodnie z własnymi upodobaniami, a dzięki intuicyjnemu interfejsowi aplikacji obsługa programu jest bardzo prosta nawet dla niezaawansowanych użytkowników. Całe oprogramowanie było tworzone w oparciu o narzędzie automatyzujące budowę „Maven", które pozwala w sposób uporządkowany zarządzać działaniem programu przez różne osoby mające styczność z kodem źródłowym programu.

8 Wnioski

Projekt "*Tanks*" był zadaniem, który pozwolił na dostrzeżenie esencji programowania obiektowego w języku „Java". Konieczność zdefiniowana w tym projekcie obiektów abstrakcyjnych, które łączą wspólne zależności i wzajemnie ze sobą współgrają umożliwiło zgłębienie koncepcji stojącej za programowaniem obiektowym i zrozumieć działanie mechanizmów z nim związanych. Dodatkowo konieczność stworzenia interfejsu graficznego unaoczniła trudności związane z tworzeniem warstwy graficznej i połączeniem jej z logicznym działaniem oprogramowania.