PROJEKT: BATTLESHIPS

Autor: Jakub Orłowski

Data oddania: 22.01.2024r.

Wersja oprogramowania: 1.0

1. Opis projektu:

Projekt ten jest stworzony jako zadanie drugie do zrealizowania na przedmiot Komunikacja Człowiek-Komputer. Treść zadania wygląda następująco:

Zadanie polega na stworzeniu aplikacji działającej w trybie graficznym. Powinna być pod względem logiki dokładnie ta sama aplikacja, co w przypadku pierwszego projektu, tylko ze zmienioną warstwą prezentacji z tekstowej na graficzną.

Oceniane będą następujące aspekty:
- sposób zmiany warstwy prezentacji z tekstowej na graficzną (z wyszczególnieniem zmian na poziomie logiki: im mniej konieczności modyfikacji, tym lepiej),
- interfejs użytkownika (wskazane użycie do obsługi zarówno myszki, jak i klawiatury, w zależności od zastosowanych kontrolek),
- animacje,
- sposób odświeżania widoku (najmniej punktowane odświeżanie całego widoku z każdą najmniejszą zmianą, wyżej odświeżanie jedynie elementów, które uległy zmianie),
- ogólne wrażenie użytkowe.

Proszę zwrócić szczególną uwagę na kwestię interfejsu użytkownika i sposobu poruszania się po aplikacji, korzystania z niej.”

Projekt: BattleShips to gra konsolowa w Javie umożliwiająca rozgrywkę w popularną grę statki w wersji tekstowej ORAZ graficznej. Pozwala ona na toczenie wojny przeciwko trzem poziomom sztucznej inteligencji oraz przeciwko innym graczom przy pomocy dedykowanego serwera.

2. Opis funkcjonalności:

- Gra przeciwko AI poziom 1 ✓
- Gra przeciwko AI poziom 2 ✓
- Gra przeciwko AI poziom 3 ✓
- Gra przeciwko graczowi w sieci ✓

3. Szczególnie interesujące zagadnienia projektowe:

V.0.0 nie edytowane

System jest stworzony na wzorcu MVC-podobnym gdzie kontroler odświeża model, a renderer go wyświetla.

W projekcie zostały użyte:

- interfejsy, które pozwalają na narzucenie wymogów klasom implementującym je, dzięki czemu jest możliwa modularność np. renderowania, która zamiast terminala w przyszłości weźmie zaimplementowany obiekt JFrame

- klasy abstrakcyjne, dzięki którym można zgrupować wspólne funkcjonalności jak zaatakowanie planszy AI, a oddzielić poszczególne implementacje takie jak ruch AI przeciwko graczowi
- pętla gry wspierająca zmianę rozmiaru ekranu, pracująca na stanach i odświeżająca ekran tylko kiedy została wykonana jakaś akcja
- singleton, którym jest silnik gry jako przykład wykorzystania wzorca projektowego
- metoda fabrykująca zwracająca poszczególne kontrolery zależnie od odebranego stanu poprzedniego kontrolera
- operacje na bitach, które były niezbędne do następnego punktu czyli:
- Wave Function Collapse – własny pomysł oraz implementacja algorytmu wykorzystującego powyższą ideę. Użyte to zostało do generowania losowego ustawienia statków na planszy AI. Załóżmy, że mamy liczbę 8-bitową. Niech jej bity oznaczają rozmiar statku: 2 od lewej to długość 4, kolejne 2 to długość 3, kolejne to 2 i kolejne to długość 1. Lewy z nich oznacza orientację poziomą a prawy pionową. Następnie jest ustawiany stan początkowy całej tablicy czyli gdzie jaki statek może zostać wstawiony. Randomizer losuje pozycję i sprawdza czy zakolejkowany statek można tam postawić poprzez operację bitową, jeśli nie to losuje nową pozycję, jeśli tak to go stawia na finałowej planszy i modyfikuje ją tak, że ustawia 0 tam gdzie jest statek i wokół niego, a po lewej i górnej stronie tej wody modyfikuje bity komórek odpowiednio do możliwych stanów. Pierwsze dwa zdjęcia poniżej (lewo, prawo) reprezentują tablicę stanów i finałową planszę na początku. Po wylosowaniu pola D7 i orientacji pionowej, zmiany ukazują następne zdjęcia.

	1	2	3	4	5	6	7	8	9	10
A	255	255	255	255	255	255	255	191	175	171
B	255	255	255	255	255	255	255	191	175	171
C	255	255	255	255	255	255	255	191	175	171
D	255	255	255	255	255	255	255	191	175	171
E	255	255	255	255	255	255	255	191	175	171
F	255	255	255	255	255	255	255	191	175	171
G	255	255	255	255	255	255	255	191	175	171
H	127	127	127	127	127	127	127	63	47	43
I	95	95	95	95	95	95	95	31	15	11
J	87	87	87	87	87	87	87	23	7	3

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

	1	2	3	4	5	6	7	8	9	10
A	255	255	255	255	255	95	95	31	175	171
B	255	255	255	255	255	87	87	23	175	171
C	255	255	191	175	171	0	0	0	175	171
D	255	255	191	175	171	0	0	0	175	171
E	255	255	191	175	171	0	0	0	175	171
F	255	255	191	175	171	0	0	0	175	171
G	255	255	191	175	171	0	0	0	175	171
H	127	127	63	47	43	0	0	0	47	43
I	95	95	95	95	95	95	95	31	15	11
J	87	87	87	87	87	87	87	23	7	3

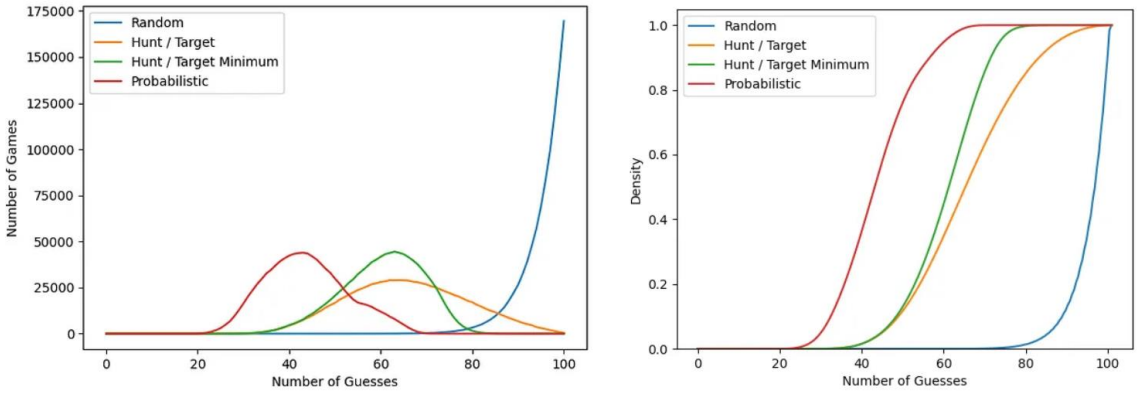
	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D							1			
E							1			
F							1			
G							1			
H										
I										
J										

V.1.0 nowe funkcjonalności:

- Aplikacja została poddana lekkiej abstrakcji pozwalającej na własne implementacje warstwy graficznej za pomocą implementacji interfejsu SpecificWindow z uwagą, że powinien tam powstać obiekt, który będzie reprezentował okno i nasłuchiwał na wciśnięcia klawiszy

klawiatury. Dzięki temu jesteśmy w stanie za pomocą argumentów programu ustalić wersję naszego okna np. podając tam wartość SWING lub LANTERNA.

- Aplikacja została poszerzona o rozgrywkę przeciwko trudniejszych wersjom botów. Poziom 2 dalej strzela losowo, natomiast przy trafieniu jest on w stanie wyszukiwać sąsiednie pola w aż do zatopienia statków. Poziom 3 dodatkowo utrudnia nam rozgrywkę poprzez zastosowanie taktyki prawdopodobieństwa. Poniższe zdjęcie przedstawia ile średnio strzałów potrzebują dane poziomy do zakończenia gry: Niebieska linia – poziom 1, żółta linia – poziom 2, czerwona linia – poziom 3.



Game length distributions for all strategies (left), cumulative distribution functions for all strategies (right). Image by author.

- Udało się utworzyć prymitywną lecz działającą wersję dedykowanego serwera, który łączy pierwszych dwóch graczy do wspólnej gry. Wymiana informacji odbywa się za pomocą Socketów, gdzie gracze otrzymują informacje o potrzebnym stanie i wysyłają własne w celu rozstrzygnięcia ataków i ustaleniu zwycięzcy. Dzięki odizolowaniu logiki od warstwy graficznej można grać ze sobą na różnych wersjach. Przykład na zdjęciu poniżej:

4. Instrukcja instalacji:

Prosimy o poinformowanie twórcy o chęci instalacji gry ze względu na prywatne repozytorium. Wtedy twórca będzie mógł dodać taką osobę w celu klonowania repozytorium.

Po dodaniu do repozytorium możemy sklonować je poprzez polecenie:

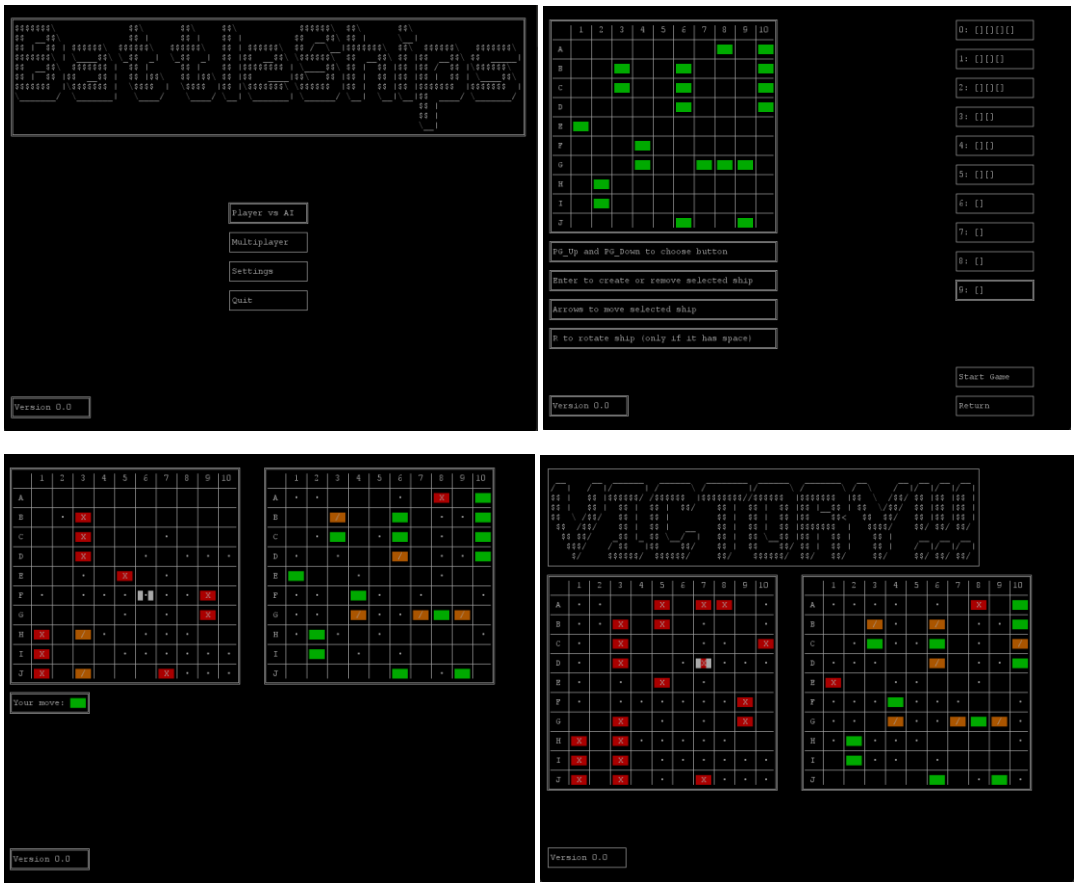
```
~> git clone https://github.com/JakiuDeus/Battleships.git
```

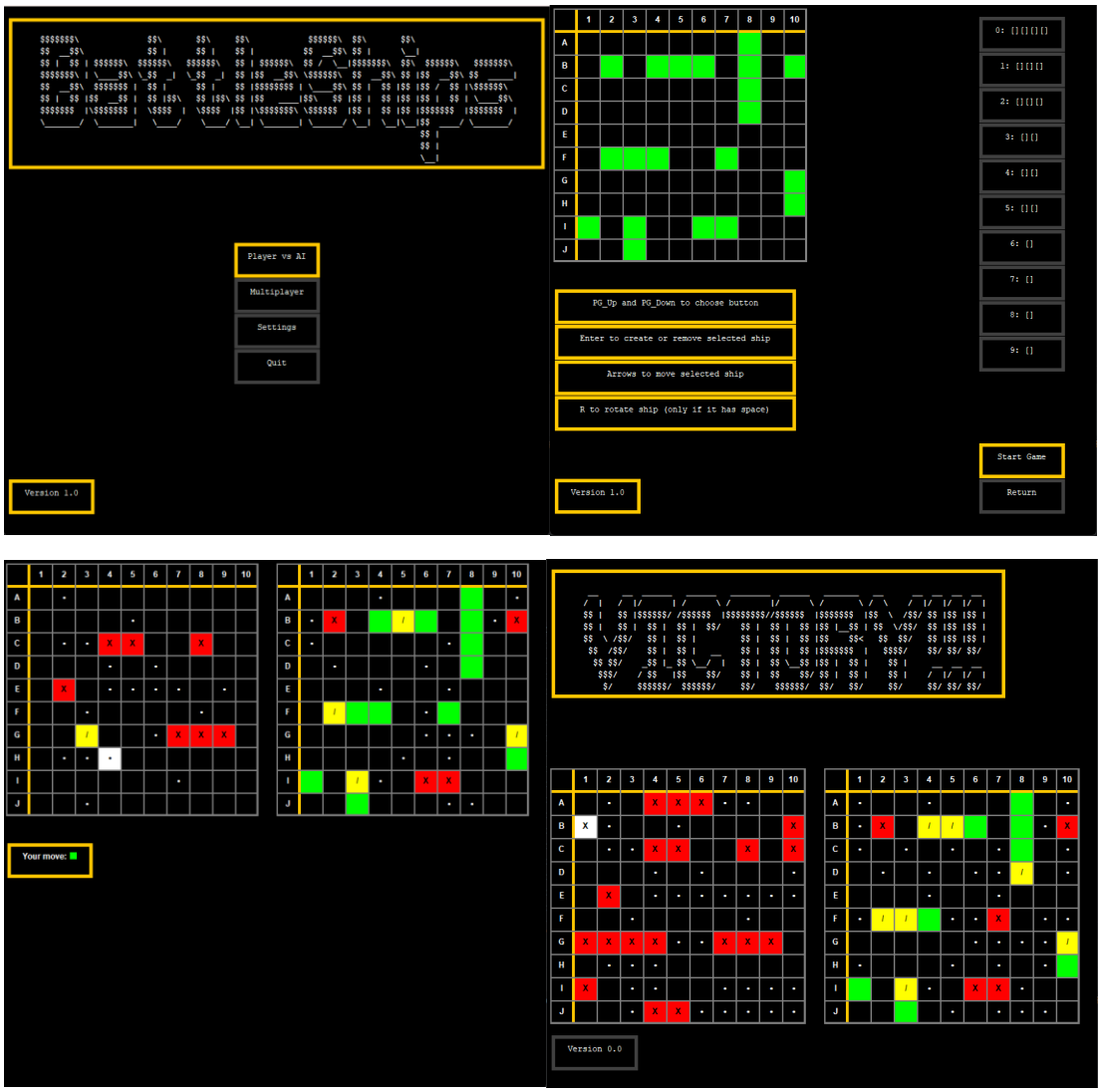
Następnie należy odpalić program serwerowy, jeśli chcemy grać przez sieć, przez środowisko programistyczne, najlepiej IntellIJ. Tak samo robimy z głównym programem. Możemy uruchomić dwie instancje na raz co oferuje taką rozgrywkę.

5. Instrukcja użytkownika:

Widoki można podzielić na cztery rodzaje:

- Widok typu Menu
- Widok budowy
- Widok gry
- Widok wyniku





Gdy jesteśmy na widoku typu Menu poruszamy się po przyciskach strzałkami do góry i w dół, a przyciskiem Enter przechodzimy do dalszej części. Wybrany przycisk różni się obwódką od niewybranych przycisków.

W widoku budowania mamy następujące akcje:

- Page_UP oraz Page_DOWN służą do poruszania się po przyciskach z prawej strony
- Enter w przypadku statków kładzie go albo zabiera z planszy.
- Strzałki służą do ruszania wybranego statku, który jest na planszy
- R – służy do obracania statku. Obrót jest możliwy gdy możemy obrócić statek trzymając go za lewy górny róg. Można to przyrównać do zegara między godzinami 3 i 6.

Gdy statki najeżdżają na siebie lub się stykają bokami bądź rogami lub gdy nie wszystkie statki zostały użyte, gra przy starcie poinformuje nas że statki nie są poprawnie rozłożone. Należy je wtedy odpowiednio poprawić aby przestrzegały te reguły.

W widoku gry mamy widoczne dwie plansze. Pierwsza jest przeciwnika i my na niej wykonujemy ruchy, a druga jest nasza i przeciwnik wykonuje na niej ruchy. Pod planszą jest oznaczenie czyj jest ruch. Jeśli jest zielone to możemy strzelać, jeśli czerwone to jest to ruch przeciwnika. AI myśli przez sekundę przed ruchem. Poziom 1 strzela losowo, poziom 2 wyłapuje statki obok, poziom 3 ma własną mapę prawdopodobieństwa, którą z każdym ruchem aktualizuje prowadząc go do najszybszej

wygranej. Gracz sieciowy przesyła przez serwer informacje o strzałach do nas a my odpowiadamy rezultatem tego strzału co owocuje w zaznaczeniu tego na obu planszach i vice-versa.

W tym widoku sterujemy strzałkami po planszy gdzie rusza się nasz kursor. Gdy pole nie zostało odgadnięte możemy kliknąć Enter. Wtedy oddamy strzał i zostaniemy poinformowani poprzez kolor i znak co się stało. Czarny i kropka oznacza pudło, Pomarańczowy i '/' oznacza trafienie ale nie zatopienie. Jak statek zostanie zatopiony zmieni się on na kolor czerwony i znaki 'X'.

W widoku wyniku widzimy obracający się napis VICTORY!!! lub DEFEAT!!! zależnie od wyniku, oraz tablice pokazujące ostatni wygląd przed zakończeniem. Po wciśnięciu Enter przejdziemy do ekranu głównego

6. Wnioski:

Odizolowanie warstwy graficznej od logiki oraz implementacja modularności warstwy prezentacyjnej narzuca niestety pewne ograniczenia jeśli chcemy wykorzystać te same komponenty. Mimo wszystko nic nie stoi na przeszkodzie aby zaimplementować własne szablony. Domyślne są możliwe do użycia przez Swing oraz Lanternę, z drobnymi usprawnieniami graficznymi w przypadku Swinga.

Komunikacja sieciowa wydawała się prosta ale tutorial do pomocy jej implementacji (która była do gry Connect 4) był przestarzały i wymagał zmniejszenia wersji Javy po stronie serwera do 15 ze względu na zmiany w 17 usuwającą metodę finalize() dla klasy Object. Mimo że powstał on w okolicach 1996 roku to jego implementacja ostatecznie odniosła sukces po długich poprawkach.

7. Samoocena:

Cała funkcjonalność dedykowana pod projekt została zaimplementowana. Mimo że wersja Swing wygląda bardzo podobnie do Lanterny to taki był tego cel, aby wykorzystać to co zostało napisane w zadaniu 1. Aby rozwiać wątpliwości trzeba ustalić że Lanterna była podzielona na tabelę znaków gdzie znak mógł mieć swoje tło. W Swingu natomiast stosujemy JPanele, które mają własne dedykowane obramówki a tekst pobierają z istniejących już komponentów. Oczywiście da się dodać specjalne wyglądy, jakieś tła, ale presja innych projektów nie umożliwia mi zaimpementowanie tej funkcjonalności na czas. Długa i męcząca była implementacja komunikacji sieciowej wymagająca mojej pełni siły do prawidłowej komunikacji między serwerem a dwoma graczami. Mam nadzieję, że dopięcie projektu w całość i umożliwienie jego modulraności poskutkuje pozytywną oceną.

Źródła:

- https://towardsdatascience.com/coding-an-intelligent-battleship-agent-bf0064a4b319 - taktyki sztucznej inteligencji w statkach, własna implementacja oparta o wyjaśnienia.
- http://www.gbengasesan.com/fyp/65/index.htm - przykład implementacji serwera do prostych gier turowych z 1996 roku, własna implementacja oparta o wyjaśnienia.