Grupa JP.GD.4
Kacper Roszczyna, Mateusz Byczkowski, Bartłomiej Przewoźniak

Szachy (PSZT 2016L) – Sprawozdanie

1. Opis tematu.

Jako projektową część przedmiotu Podstawy Sztucznej Inteligencji nasza grupa dostała zadanie zaprojektowania oraz zrealizowania aplikacji grającej w szachy (wykorzystującej algorytm Min-Max).

2. Cel i założenia.

Do realizacji projektu zostanie wykorzystany język programowania Java. GUI zostanie zrealizowane za pomocą pakietów SWING. Jako środowisko programistyczne wybraliśmy Eclipse (wersje od 4.0 wzwyż).
Zaimplementowane będą trzy tryby rozgrywki:

- Komputer (algorytm losowy) vs. Komputer (algorytm min-max)
- Człowiek vs. Komputer (algorytm min-max)
- Człowiek vs. Komputer (algorytm losowy)

W przypadku pierwszego trybu będzie to prezentacja skuteczności zaimplementowanego algorytmu w porównaniu do algorytmu losowego.

Celem projektu jest uzyskanie algorytmu, który będzie grał lepiej niż algorytm losowy, czyli w trybie Komputer vs. Komputer pokona zawsze algorytm losowy. Celem projektu, jest też uzyskać algorytm, który będzie w stanie pokonać człowieka, który jest przeciętnym graczem w szachy. Kolejnym kryterium projektu jest to, że czas wykonania algorytmu będzie możliwie krótki, czyli z punktu widzenia gracza komputer podejmie decyzję niemal natychmiastowo.
Jako rodzaj szachów, który będziemy implementowali wybraliśmy szachy szybkie (60 minut na partię oraz 60 sekund na wykonanie ruchu).

3. Plan projektu.

Zakładamy wykorzystanie algorytmu min-max. Zakładamy też wykorzystanie wzorca MVC celem lepszego rozdziału kontroli. Będziemy starać się wykorzystać metodologię TDD:

„1. Najpierw programista pisze automatyczny test sprawdzający dodawaną funkcjonalność. Test w tym momencie nie powinien się udać.
2.Później następuje implementacja funkcjonalności. W tym momencie wcześniej napisany test powinien się udać.
3. W ostatnim kroku programista dokonuje refaktoryzacji napisanego kodu, żeby spełniał on oczekiwane standardy.”

źródlo: https://pl.wikipedia.org/wiki/Test-driven_development

Będziemy stosować praktyki z Czystego Kodu Robert. C. Martine'a.

Na projekt składają się następujące elementy, nad którymi praca będzie przebiegała równolegle:

1. Utworzenie GUI i implementacja funkcji GUI.
2. Zaimplementowanie podstawowych elementów rozgrywki, przesuwania figur, wyboru stron gry.
3. Zaimplementowanie warunków kończących rozgrywkę(mat) oraz warunku specjalnego (szach)
4. Implementacja algorytmu Min-Max.
5. Implementacja frameworku testowego porównującego algorytmy.

W projekcie zastosujemy takie metody pisania kodu, aby można było użyć dowolnego algorytmu, do wykonania testów, który na wejściu będzie przyjmował planszę z umieszczonymi figurami oraz na wyjściu podawał planszę oraz ruch, który został wykonany.

4. Opis użytego algorytmu w projekcie

Algorytm MinMax dla każdego możliwego ruchu w danej sytuacji na szachownicy oblicza jego wartość(w oparciu o przypisane wartości dla poszczególnych figur), a następnie wybiera najlepsze rozwiązanie. Wartość ruchu zależy przede wszystkim od możliwości zbicia figury (oceny według standardów szachowych) oraz możliwości zagrożenia królowi przeciwnika (najwyższa ocena). W zależności od wybranego poziomu zagłębienia dla algorymu MinMax analizujemy odpowiednią liczbę kolejnych ruchów zakładając, że przeciwnika zawsze będzie grał maksymalnie na naszą niekorzyść. Optymalnym poziomem zagłębiania okazał się poziom czwarty(przewidywanie 4 ruchów do przodu) ze względu na płynność obliczeń programu oraz zadowalające działanie algorytmu.

Algorytm przyjmuje strategię ofensywną – w przypadku wymiany równoważnych figur – decyduje się na bicie. Algorytm potrafi dobrze obronić się przed matem i również w pierwszej kolejności dąży do zamatowania przeciwnika.

5. Opis aplikacji GUI wraz z instrukcją użytkownika

Po uruchomieniu aplikacji użytkownik ma do wyboru 3 opcje (poprzez wpisanie odpowiedniej liczby w konsoli tekstowej i zatwierdzenie [enter]):

1) Obserwowanie pojedynku algorytmu losowego(kolor biały) z algorytmem min-max (kolor czarny).
2) Rozegranie partii z algorytmem min-max (użytkownik – kolor biały, min-max – kolor czarny).
3) Rozegranie partii z algorytmem losowym (użytkownik – kolor biały, losowy – kolor czarny).

Po prawej stronie okna aplikacji znajdują się zegary pokazujące i odmierzające odpowiednio:
- czas do końca rozgrywki (60 minut na partię)
- czas do końca tury (czas na wykonanie pojedynczego ruchu: 60 sekund)

Wszystkie komunikaty dotyczące przebiegu rozgrywki są wypisywane w konsoli tekstowej. Możliwe są następujące komunikaty:
- szachowanie króla białego
- szachowanie króla czarnego
- szach mat (wygrana odpowiedniego gracza)
- przekroczenie czasu na turę (przegrana odpowiedniego gracza)
- przekroczenie czasu na rozgrywkę (remis)

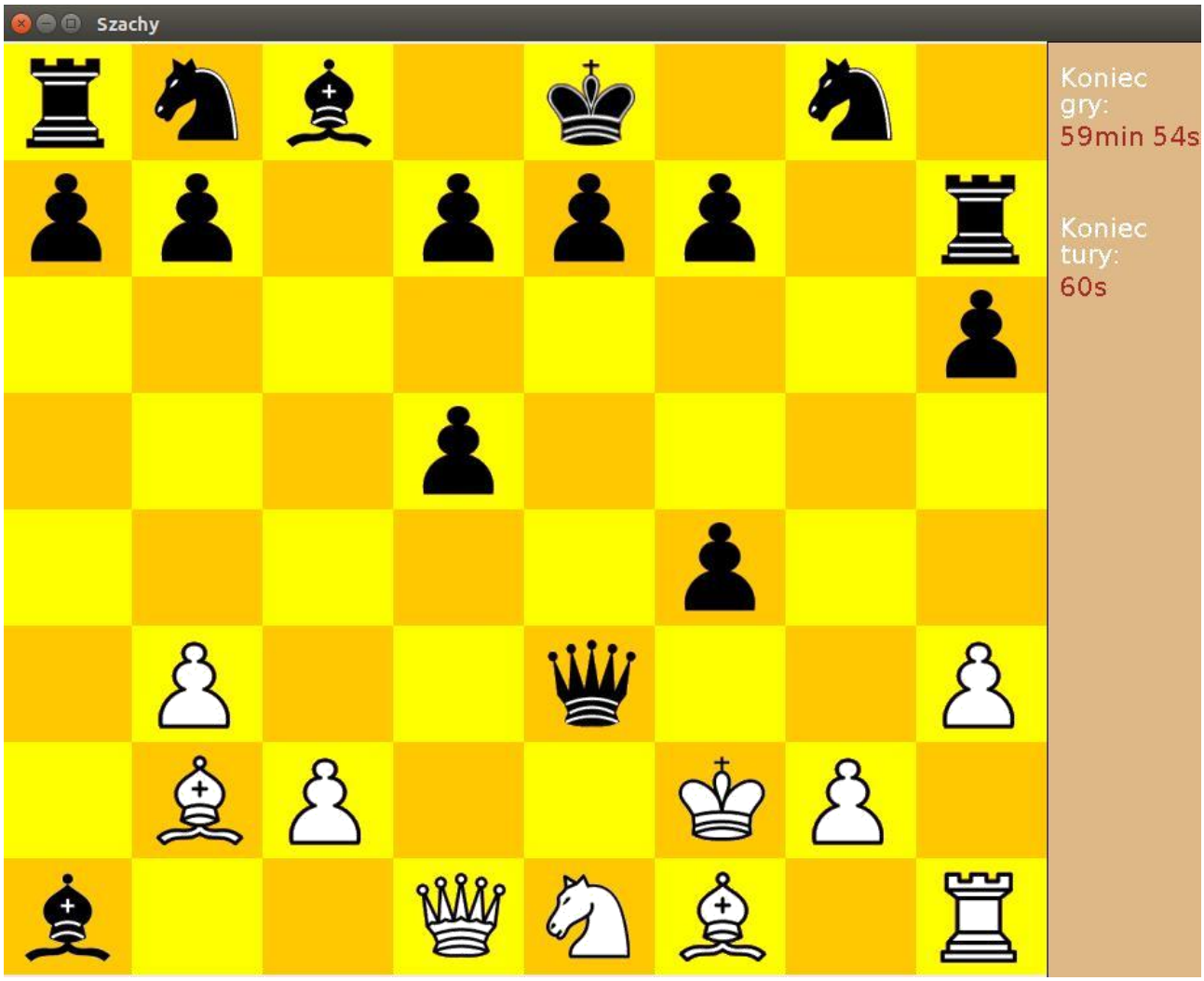
6. Testowanie aplikacji i analiza przykładowych wyników

Do testowania użyliśmy pierwszego trybu rozgrywki (losowy vs min-max) oraz rozgrywaliśmy partie własnoręcznie z algorytmem min-max. W przypadku pojedynków z algorytmem losowym algorytm min-max zawsze osiąga przewagę punktową. W około 80% udaje mu się doprowadzić do mata, natomiast czasami nie udawało się tego dokonać ze względu na brak zaimplementowanej strategii szachującej dla konkretnych przypadków (przewidywanie 4 ruchów do przodu okazywało się niewystarczające).

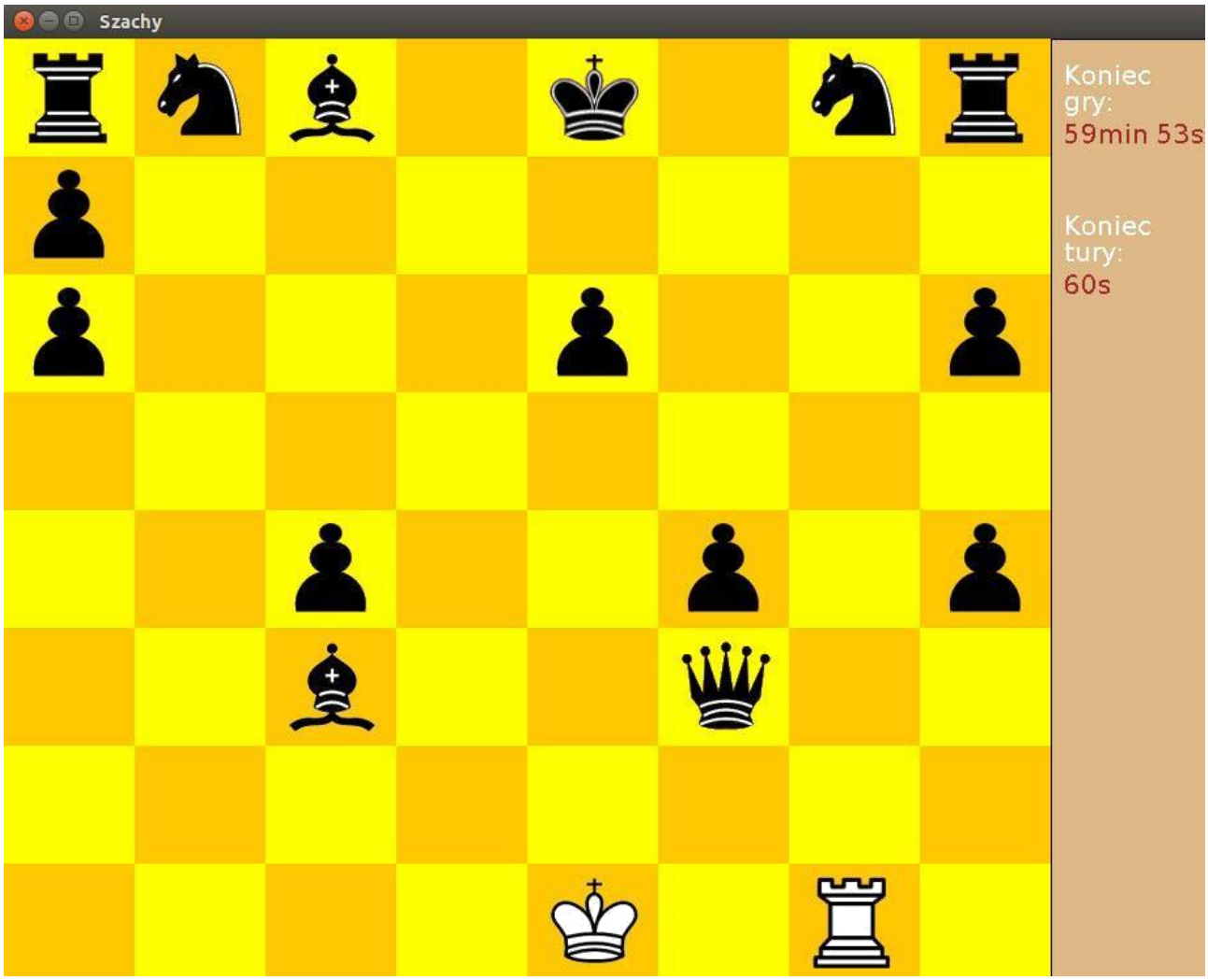
W przypadku gry z mało doświadczonym/średnio-zaawansowanym graczem algorytm min-max również w większości przypadków osiągał przewagę nad przeciwnikiem – jednak czasami mimo zdecydowanej przewagi nie udawało mu się zamatować przeciwnika ze względu na brak opracowanej strategii szachującej konkretnymi figurami.

Przykładowe screeny z gry algorytmu min-max(czarne) z algorytmem losowym(białe):

wygrana po 4 sekundach



wygrana po 6 sekundach



wygrana po 7 sekundach

W przedstawionych przykładach widać, że algorytm min-max w pierwszej kolejności dąży do zamatowania przeciwnika. Jednak w niektórych przypadkach analizowanie 4 ruchów w przód okazywało się niewystarczające do zamatowania przeciwnika.

7. Podsumowanie i wnioski.

Kluczowe decyzje projektowe:

• do realizacji projektu został wykorzystany język programowania Java

• GUI zostało zrealizowane za pomocą pakietu SWING

• zaimplementowane zostały trzy tryby rozgrywki: - człowiek vs algorytm losowy
- człowiek vs algorytm min-max
- algorytm losowy vs algorytm min-max

• zdecydowaliśmy się zrealizować projekt w oparciu o wzorzec MVC – uznaliśmy że takie podejście pozwoli na sprawne połączenie pracy członków zespołu i zapewni przejrzystość rozwiązania

• stosowaliśmy praktyki z „Czystego Kodu" Robert. C. Martine'a.

• zaimplementowaliśmy podstawowych elementów rozgrywki, przesuwania figur według zadanych reguł, wyboru stron gry

• zaimplementowanie warunku kończącego rozgrywkę(mat) oraz wymuszenia obowiązkowej obrony przed matem

• zaimplementowanie algorytmu losowego oraz algorytmu min-max

Opis struktury programu:

Program był realizowany w oparciu o wzorzec MVC.

- Model – moduł przechowujący aktualny stan rozgrywki (dane) oraz reguły gry
- Kontroler – moduł algorytmiczny oraz moduł kontrolerów (odpowiadający za wykonywanie odpowiednich ruchów na przemian przez graczy oraz uaktualnianie widoku)
- Widok – moduł wyświetlający stan rozgrywki oraz zbierający polecenia wykonania ruchu od użytkownika

Wnioski dotyczące osiągniętych rezultatów:

Algorytm MinMax dla każdego możliwego ruchu w danej sytuacji na szachownicy oblicza jego wartość, a następnie wybiera najlepsze rozwiązanie. Wartość ruchu zależy przede wszystkim od możliwości zbicia figury (oceny według standardów szachowych) oraz możliwości zagrożenia królowi przeciwnika (najwyższa ocena). W zależności od wybranego poziomu zagłębienia dla algorymu MinMax analizujemy odpowiednią liczbę kolejnych ruchów zakładając, że przeciwnika zawsze będzie grał maksymalnie na naszą niekorzyść. Optymalnym poziomem zagłębiania okazał się poziom czwarty ze względu na płynność obliczeń programu oraz stosunkowo zachowania przeciwnika.

Algorytm przyjmuje strategię ofensywną – w przypadku wymiany równoważnych figur – decyduje się na bicie. Algorytm potrafi dobrze obronić się przed matem i również w pierwszej kolejności dąży do matu.

Zrealizowany algorytm min-max zawsze osiągał przewagę nad algorytmem losowym oraz prawie zawsze nad użytkownikiem aplikacji. Jednak w niektórych przypadkach mimo przewagi nie był wstanie zamatować przeciwnika, ponieważ analiza 4 ruchów w przód okazywała się niewystarczająca.

Usprawnieniem byłoby opracowanie różnych strategii otwarć rozgrywki oraz opracowanie strategii prowadzącej do mata dla konkretnych sytuacji (np. szachowanie 2 gońcami i królem).

Szachy okazały się być złożonym projektem ze względu na dość skomplikowane reguły gry (6 różnych figur o różnym sposobie poruszania się). Większą część czasu realizowania projektu zajęło samo przygotowanie aplikacji umożliwiającej grę użytkownika z algorytmem losowym.