Dokumentacja projektu zaliczeniowego z Podstaw Informatyki i Programowania (PIPR) – gra w „Neutron”

Michał Kolankiewicz, grupa 101, nr albumu 303765, prowadzący: Agnieszka Szmurło

1. Temat projektu:

„Należy stworzyć program umożliwiający rozgrywanie partii w grze planszowej neutron między człowiekiem a graczem komputerowym.

Należy stworzyć co najmniej dwa rodzaje graczy komputerowych:

1. wybierający w każdej turze losowo jeden z możliwych do wykonania ruchów
2. wybierający w każdej turze najlepszy ruch według pewnych prostych kryteriów”

2. Realizacja:

Program działa w oparciu o modyfikację listy list, która stanowi planszę 7x7 (gra odbywa się na centralnych 25 kratkach, pozostałe służą jako bandy) na której odbywa się gra. Program obsługuje funkcje, które zamieniają odpowiednie elementy list miejscami, sprawdzają możliwość wykonania ruchu, wynajdują koordynaty szukanego pionka, ewaluują pozycję, czy np. neutron znajduje się na początkowej rance, czy padł pat. Obsługuje menu, które pozwala poznać zasady gry, sterowania oraz wybranie trybu gry: z innym graczem, z komputerem na generującym ruchy losowe lub komputerem, który dokonuje podstawowej ewaluacji pozycji na podstawie tego, jaki ruch neutrona wygrywa, jaki ruch neutrona nie przegrywa, jaki ruch pionka wygrywa i jaki ruch pionka zwiększa szansę na wygraną w następnej turze (zwalnia pole na rance domowej).

3. Wprowadzone funkcje:

a) obsługa interfejsu:
   - **def show(board)**: przedstawia aktualny stan planszy (listy) „board”,
   - **def find_pawn_Y(pawn, board)**: wyszukuje numer wiersza, w jakim znajduje się szukany pionek i zwraca ten numer,
   - **def find_pawn_X(pawn, board)**: działa analogicznie do powyższego, zwraca kolumnę,
   - **def find_pawn(pawn, board):** połączenie dwóch powyższych, zwraca koordynaty pionka,
   - **def check_for_collision(pawn, direction, board):** sprawdza maksymalną odległość, jaką pionek może przebyć w wybranym kierunku i zwraca tą odległość,
   - **def move(pawn, direction, board):** przemieszcza pionek w wybranym kierunku na maksymalną odległość w oparciu o funkcję **check_for_collision**

b) elementy gry:
   - **def main_menu():** wyświetla menu opcji, które może wybrać gracz: może rozpocząć rozgrywkę, poznać zasady gry lub zasady sterowania,
   - **def gameplay():** określa kolejność i reguły, według których odgrywa się rzeczywista część gry: kolejki zawodników oraz przebieg ruchu,

- **def players_neutron_turn(board):** pozwala wprowadzić ruch neutrona w wybranym kierunku,
- **def players_pawn_turn(board):** jak wyżej, ale wybranego pionka,

c) warunki logiczne:
- **def check_for_stalemate(board):** sprawdza, czy na planszy padł pat poprzez znalezienie neutrona i sprawdzenie, czy może wykonać jakikolwiek ruch,
- **def check_for_win(board):** sprawdza, czy neutron znajduje się na rance domowej białych i zwraca odpowiedniego boola,
- **def check_for_lose(board):** jak wyżej, ale w rance domowej czarnych,

d) silniki komputerowe:
- **def randomly_generated_move(board):** wybiera losowy pionek z dostępnych i sprawdza, czy ruch dla niego jest wykonalny w oparciu o **check_for_collision**. Jeśli ruch jest niewykonalny (o długości 0), to szuka dalej możliwego ruchu, aż nie znajdzie takiego o niezerowej długości,
- **def randomly_generated_neutron_move(board):** jak wyżej, ale tylko dla neutrona,
- **def try_winning_neutron_move(board):** sprawdza, czy możliwy jest taki ruch neutronem, aby na planszy padła wygrana czarnych i jeśli taki jest, wykonuje go,
- **def try_not_losing_neutron_move(board):** jak wyżej, ale aby nie padła wygrana dla białych,
- **def try_winning_pawn_move(board):** sprawdza, czy możliwy jest taki ruch dowolnym pionkiem, aby na planszy padł pat (wygrana czarnych) i wykonuje go, jeśli taki jest,
- **def priority_pawn_move(board):** sprawdza, czy w domowej rance są jeszcze nieruszone pionki i wykonuje nimi losowy ruch

4. Podsumowanie:

Podsumowując, mogę powiedzieć, że jestem całkiem zadowolony z efektu, który udało mi się osiągnąć w tym projekcie. Jestem jednak świadomy, że nie jest to program idealny i wymagałby doszlifowania, którego nie umiem dokonać, głównie w zakresie opracowywania wyjątków. Starałem się to obejść projektując funkcje w taki sposób, by do wyjątków nigdy nie musiało dojść, stąd w kilku miejscach warunek z komunikatem „Coś poszło nie tak” i np. ponowienie pętli. Co ciekawego zauważyłem, to podczas przeprowadzania testów jednostkowych jedyne funkcje, których test nie chciał przepuścić, były z zakresu obsługi silnika robiącego ruchy nielosowe. Ciekawe jest to dlatego, że w trakcie rzeczywistego grania, silnik ten zdaje się wykonywać polecenia poprawnie, gdy ustawić pozycję tak, że rzeczywiście można ją wygrać (z pozycji silnika). Znajduje ruchy wygrywające, paty, stawia priorytet na nierozwinięte piony, mimo wszystko testy są uznawane za sprzeczne.

W pliku głównym „neutron.py” oprócz defów umieszczonych jest już 5 komend: 4 razy komenda show() oraz komenda main_menu(). Komendy show() służą pokazaniu, jakich tablic dokładnie użyłem podczas tych ostatnich 4 testów, a komenda main_menu() służy uruchomieniu gry. Wszelkie instrukcje odnośnie sterowania i zasad gry są do przeczytania w tym właśnie menu po uruchomieniu programu.

Mam nadzieję, że udało mi się spełnić jak najwięcej oczekiwań względem tego projektu oraz życzę miłego zapoznawania się z jego treścią.