Praca domowa II—Wielka Strategia

28 października 2025

1 Instrukcja

Twoim zadaniem jest napisanie gry strategicznej z gatunku Wielkiej Strategii (*Grand Strategy*). Gracz w trakcie rozgrywki pełni rolę przywódcy państwa lub tworu państwopodobnego (może to być konkretna osoba, może nie być). Państwo znajduje się na mapie (wczytywanej z dysku lub losowo generowanej) złożonej z prowincji, które należą do różnych państw. Państwa inne niż państwo gracza kierowane są przez sztuczną inteligencję lub innych graczy przez sieć (patrz sekcja 4). Państwa mają skarbiec i armię. Mogą toczyć między sobą wojny i prowadzić prostą dyplomację. Rozgrywka toczy się turowo lub w czasie rzeczywistym. Gra jest piaskownicą (*Sandbox*). Cel gracz obiera sobie sam. Całość zrealizowana jest w postaci aplikacji graficznej lub aplikacji TUI. Grę zaimplementować można z minimalnym użyciem zewnętrznych bibliotek, lub użyć można gotowego frameworku lub silnika (np. `bevy`).

Wzorować można się na grach takich jak:

- Europa Universalis,
- Crusader Kings,
- Victoria,
- Hearts of Iron,
- seria Total War.

Termin projektu: 16.01.2026. Każdy tydzień opóźnienia skutkuje ograniczeniem maksymalnej liczby punktów o 5.

W sekcji 3 znajdziesz wymaganą do zaimplementowania funkcjonalność. Za jej pełną implementację otrzymać możesz 20pkt. W sekcji 4 znajdziesz dodatkową funkcjonalność. Przy każdej funkcjonalności znajduje się liczba dodatkowych punktów za jej implementację i jej identyfikator. Aby otrzymać punkty za dodatkową funkcjonalność, musisz zaimplementować funkcjonalność z sekcji 3 za przynajmniej 15 punktów. W sekcji 5 znajdziesz rzeczy, których nie możesz robić w projekcie.

Jeśli w poleceniu jest napisane, że należy zaimplementować strukturę/cechę/funkcję o nazwie `X`, która nie jest jednak definiowana szczegółowo przez polecenie, to jej implementacja jest dowolna.

Dozwolone jest używanie dowolnych zewnętrznych bibliotek. Dozwolone jest używanie dowolnych elementów z biblioteki standardowej, o ile nie są one zakazane przez sekcję 5.

Ocena za projekt liczona jest ze wzoru

$$k = \min(r + a + p, 40),$$

gdzie k—ocena końcowa, r—punkty z sekcji 3, $a = 0$ jeśli $r < 15$, lub a jest równe punkty z sekcji 4 jeśli $r \geq 15$, p—punkty z sekcji 5.

2 Biblioteki

W implementacji mogą przydać się następujące biblioteki. Nie są to jedyne dozwolone biblioteki, nie trzeba też używać którejkolwiek z nich.

1. Grafika
   (a) `ratatui`—biblioteka do tworzenia interfejsów TUI.
   (b) `console_engine`—biblioteka ułatwiająca rysowanie w konsoli.
   (c) `macroquad`—prosta biblioteka do tworzenia gier 2D i 3D.
   (d) `ggez`—prosty framework do tworzenia dwuwymiarowych gier.
   (e) `bevy`—rozbudowany silniki z systemem ECS i renderowaniem grafiki 2D lub 3D.
2. Przydatne w implementacji rozgrywki
   (a) `specs`—system ECS niezależny od silnika Bevy.
   (b) `pathfinding`—implementacja różnych algorytmów wyszukiwania ścieżek i przepływów.
   (c) `noise`—biblioteka do generowania szumu.
3. Inne
   (a) `anyhow`—biblioteka do obsługiwania błędów w aplikacjach. Lepiej nadaje się do tego projektu niż `thiserror`.
   (b) `serde`—najczęściej używana biblioteka do serializacji JSON, YAML, TOML, etc.
   (c) `parking_lot`—wydajniejsze implementacje prymitywów synchronizacyjnych.
   (d) `assets_manager`—manager zasobów.

3 Wymagana funkcjonalność

1. Rozgrywka
   (a) Mapa
   - (2pkt) wczytywana z pliku lub generowana losowo,
   - (1pkt) która składa się z prowincji
   - (1pkt) i typów terenu po jednym na prowincję, wymagane są przynajmniej: płaski, las, góry, woda. Prowincje z terenem wody nie należą do państw i nie można po nich poruszać jednostek. Jeśli umiejscowienie gry (np. przestrzeń kosmiczna) sprawia, że te typy terenu są bez sensu, można je zastąpić innymi.

   (b) Państwa, które mają
   - (1pkt) flagę lub herb oraz przypisany kolor,
   - (1pkt) kontrolowane prowincje,
   - (1pkt) armię—w prowincjach stacjonować mogą jednostki, które mogą się między nimi przemieszczać,
   - (1pkt) pieniądze w skarbcu, które można wydawać na tworzenie jednostek i budowę budynków w prowincjach (budynki mogą np. pozwalać na rekrutację jednostek lub zwiększać dochód z prowincji—dowolnie).
   - (2pkt) kontrolowane mogą być przez prostą sztuczną inteligencję, która według dowolnych heurystyk tworzy armię, buduje budynki, próbuje poszerzać granice państwa i toczyć wojny.

   (c) Państwa mogą też

- (1pkt) prowadzić wojnę z innymi państwami. Wtedy ich jednostki mogą okupować wrogie prowincje i toczyć bitwy z jednostkami przeciwnika. W trakcie bitwy porównywane są parametry obu armii i na tej podstawie rozstrzygana jest walka.
- (1pkt) Prowadzić prostą dyplomację z innymi państwami: wypowiadać im wojny i zawierać pokój.

(d) (1pkt) Rozgrywka jest turowa lub w czasie rzeczywistym W przypadku rozgrywki turowej istnieje przycisk rozpoczęcia kolejnej tury. W przypadku rozgrywki w czasie rzeczywistym istnieje możliwość zmiany szybkości upływu czasu.

2. Grafika i interfejs

   (a) (2pkt) Gra ma interfejs pozwalający toczyć rozgrywkę.

   (b) (1pkt) Mapa może być wyświetlona w trybie przynależności do państw (każda prowincja pokolorowana kolorem państwa, do którego należy) i w trybie terenu (każda prowincja pokolorowana ze względu na typ terenu)

   (c) (1pkt) Na mapie wyświetlane są armie.

   (d) (1pkt) Menu gry pozwalające na wczytanie gry, rozpoczęcie nowej gry z wyborem państwa, ew. opcje gry.

3. Pozostałe

   (a) (1pkt) Wątki gry i wątek rysowania powinny być oddzielne. Jeśli framework lub silnik którego używasz czyni to bardzo problematycznym, to zamiast tego w osobnym wątku zachodzić ma zapis stanu gry do pliku, z informacją o zakończeniu zapisu wyświetlaną w grze. Logika gry nie może jednak nigdy blokować renderingu—gra nie powinna się zacinać, gdy wykonywane jest wiele obliczeń.

   (b) (1pkt) Grę można zapisać do pliku i wczytać ją aby kontynuować rozgrywkę.

4 Dodatkowa funkcjonalność

Mechaniki gry:

1. (`population`, 5pkt) Zaawansowana symulacja populacji. Każda prowincja zawiera informacje o liczbie ludności, jej kulturze, warstwie społecznej. Populacja zmienia się dynamicznie w trakcie gry. Przemarsz wrogiej armii powoduje jej spadek, budynki w prowincji wpływają na jej rozwój.

2. (`diplomacy`, 5pkt) Zaawansowana dyplomacja między państwami. Powinna być przynajmniej możliwość: zawierania sojuszy, negocjacji pokojowych (przekazywanie prowincji, uczynienie państwa podległym innemu, płacenie trybutu/reparacji), sprawdzenia relacji między państwami (mierzone liczbą 0-100 lub w bardziej zaawansowany sposób), które wpływają na szansę powodzenia zawierania sojuszy i akceptacji wezwania sojusznika do wojny.

3. (`technology`, 5pkt) System drzewa technologii—nowsze technologie pozwalają na budowę lepszych budynków, lepszych jednostek, używanie nowych opcji dyplomatycznych i potencjalnie innych rzeczy.

4. (`economy`, 5pkt) Złożony system ekonomii. Handel ze zmiennymi cenami towarów zależnymi od popytu i podaży. Budowa budynków i rekrutacja jednostek wymaga konkretnych towarów. System zaciągania pożyczek, inflacji i bankructwa państwa.

5. (`stock_exchange`, wymaga `economy`, 10pkt) Rozszerzenie systemu ekonomii o giełdę. System spółek, które handlują konkretnymi towarami i czerpią z tego zysk (mogą też produkować/konsumować towary i je sprzedawać/kupować). Giełda, na której państwa mogą kupować akcje

spółek, które wypłacają dywidendy. Ceny akcji zmieniają się w zależności od popytu i podaży na akcje. Handel odbywa się tylko przez spółki. Możliwość handlu w derywatach: call options, put options, futures. Emisja obligacji przez państwa i spółki. Spółki można zaimplementować jako państwa (bez lub z prowincjami) lub inne byty, do wyboru.

6. (`warfare`, 5pkt) Rozbudowany system bitew. Bitwy trwają kilka tur lub ticków. Jednostki mają różne parametry, które wpływają na przebieg bitwy. Po rozpoczęciu bitwy, dołączyć mogą do niej dodatkowe jednostki, jeśli przejdą do prowincji z bitwą. Przebieg bitwy jest po części losowy. Teren wpływa na wynik—jeśli armia atakuje inną armię, która stacjonuje w terenie górzystym, to pierwsza otrzymuje w pewien sposób przewagę.

7. (`events`, 5pkt) System losowych wydarzeń i modyfikatorów. Wydarzenie składa się z opisu oraz kilku opcji, które może wybrać gracz. Każda z nich nakłada na państwo inne modyfikatory, dodaje lub odejmuje fundusze etc. Zaimplementowane powinno być też wsparcie dla wydarzeń warunkowych, np. wydarzenie „Susza"może pojawić się tylko latem, a wydarzenie „Niskie morale"tylko w trakcie wojny.

8. (`navy`, 3pkt) Możliwość budowy statków, umieszczania na nich jednostek i transportu przez prowincje wodne.

Pozostałe:

1. (`advanced_threading`, 5pkt) wątek logiki gry podzielony jest na kilka wątków. Wątki używają algorytmu *work stealing*, aby wykonywać wszystkie obliczenia potrzebne do przejścia do kolejnej tury lub ticku (ruchy ai, obliczanie dochodów, wyników bitew itd.).

2. (`replay_viewer`, 5pkt) Wbudowany system odtwarzania powtórek, pozwalający wyświetlać jak przynależność prowincji zmieniała się w trakcie rozgrywki.

3. (`map_editor`, 8pkt) Wbudowany edytor mapy, pozwalający na rysowanie terenu, granic prowincji, tworzenie państw i przypisywanie im prowincji.

4. (`audio`, 3pkt) Naciskanie na przyciski w interfejsie wydaje dźwięki. Wydarzenia takie jak wypowiedzenie wojny, zawarcie pokoju, bitwa, wydają różne dźwięki.

5. (`neural_ai`, 20pkt) Sztuczna inteligencja zaimplementowana jest poprzez sieci neuronowe z użyciem biblioteki `burn`.

6. (`multiplayer`, 10pkt + dodatkowe 25% punktów z części mechaniki gry) Gra może rozgrywać się między wieloma graczami poprzez sieć, gdzie każdy gracz przejmuje kontrolę nad innym państwem. Pozostałe państwa sterowane są przez AI.

7. (`2d`, 5pkt) Cała gra renderowana jest w grafice 2-wymiarowej (nie w terminalu).

8. (`3d`, 10pkt) Renderowana grafika jest trójwymiarowa. W każdej prowincji wyświetlany jest trójwymiarowy model miasta, armie są też trójwymiarowymi obiektami na mapie. Ukształtowanie terenu jest też widoczne w trójwymiarze.

Można też samemu wymyślić dodatkową funkcjonalność. Zaleca się jednak jej konsultację przed oddaniem projektu. Można za nią uzyskać do 20 punktów.

5 Praktyki zakazane

Pojawienie się któregokolwiek z poniższych elementów w projekcie skutkuje odjęciem tylu punktów, ile sprecyzowano.

1. Użycie metod `unwrap`, `expect` lub makra `panic`. −1pkt za 1 lub 2 użycia, −2pkt za więcej użyć. Dozwolone jednak używanie w testach.

2. Używanie struktury, gdzie właściwy byłby `enum`, to jest trzymanie w strukturze pól, których wartość czasami ma, a czasami nie ma znaczenia, np.

```
struct IP {
v4: String,
v6: String,
is_v4: bool
}
```

   -1pkt za jedno wystąpienie, -2pkt za więcej.

3. Ignorowanie błędów, kiedy powinny być obsłużone, np. wołanie funkcji, która zwraca `Result<T, E>` i sprawdzanie przez `if let` tylko przypadku `Ok`, kiedy błąd ma znaczenie (jeśli nie ma, to oczywiście dozwolone). -1pkt za jedno lub 2 wystąpienia, -2pkt za więcej.

4. Funkcje dłuższe niż 30 linijek. -1pkt za 1 lub 2, -2pkt za więcej. Funkcje takie są dozwolone w testach.

5. Funkcje, które niepotrzebnie przejmują własność obiektów, kiedy mogłyby je tylko pożyczać. -1pkt za 1-3 takich funkcji. -2pkt za więcej.

6. Używanie `clone`, kiedy jest niepotrzebny (tzn. kod kompiluje się po jego usunięciu, a nie wpływa to na logikę). -1pkt za jedno lub więcej użyć.