Programowanie współbieżne - Projekt 2017/2018

Celem projektu jest stworzenie trzyosobowej gry strategicznej, opartej o mechanizmy komunikacji między procesami systemu Linux, poznane na zajęciach laboratoryjnych z Programowania Współbieżnego.
Gra realizować ma zdefiniowane poniżej zasady rozgrywki oraz założenia architektoniczne. Ocenie nie podlega szata graficzna gry, a odpowiedzenie wykorzystanie mechanizmów systemu Linux.

ZASADY GRY

W grze udział bierze trzech graczy. Podczas rozgrywki gromadzą oni surowce, produkują jednostki oraz atakują przeciwnika. Celem gry jest wykonanie pięciu skutecznych ataków na przeciwnika.

JEDNOSTKI

Do dyspozycji graczy są cztery rodzaje jednostek:

- lekka piechota,
- ciężka piechota,
- jazda,
- robotnicy.

Każda jednostka ma określoną cenę, siłę ataku, zdolności defensywne oraz czas produkcji, określone przez tabelę 1.
Jednostki tworzone są przez gracza w ramach dostępnych w danym momencie surowców. Aby wytrenować jednostkę, gracz wydaje komendę, w której określa liczbę i rodzaj wojsk. Jednostka staje do dyspozycji gracza zaraz po wybudowaniu, nie czekając na zakończenie całego zadania — po pięciu sekundach

	Cena	Atak	Obrona	Czas produkcji
Lekka piechota	100	1	1.2	2s
Ciężka piechota	250	1.5	3	3s
Jazda	550	3.5	1.2	5s
Robotnicy	150	0	0	2s

Tablica 1: Współczynniki jednostek

od zlecenia budowy 4 jednostek lekkiej piechoty, gracz może już dysponować dwiema jednostkami.

Gracz może wydać kolejne polecenie budowy żołnierzy przed zakończeniem poprzedniego polecenia, ale nowy proces treningu rozpocznie się dopiero po zakończeniu aktualnego zadania. Nie ma możliwości anulowania raz zleconego treningu.
Aplikacja nie może zezwolić na budowę jednostek, których koszt przekracza aktualny stan surowców gracza.

SUROWCE

Zbieranie surowców odbywa się automatycznie, w tempie 50 jednostek na sekundę. Każdy wybudowany robotnik podnosi tempo wydobycia o 5 jednostek na sekundę.
Gracz rozpoczyna grę posiadając 300 jednostek surowca. Zlecenie treningu jednostek zmniejsza pulę dostępnych surowców od razu o całą kwotę konieczną do tego treningu.

WALKA

Gracz w dowolnym momencie może zdecydować się na wydanie komendy ataku. W tym celu definiuje liczbę i rodzaj jednostek biorących w nim udział. W każdym ataku mogą uczestniczyć wszystkie typy jednostek, z wyjątkiem robotników. Jednostki wysłane do ataku nie uczestniczą w ewentualnej obronie, jeżeli przeciwnik wyśle swoje jednostki w tym samym czasie.
Każdy atak, niezależnie od użytych jednostek, trwa 5 sekund. Po tym czasie uznaje się, że ocalałe jednostki znów gotowe są do obrony.
Atak uznaje się za udany, jeżeli suma siły ataku wszystkich jednostek atakujących przewyższa zdolności obronne jednostek dostępnych w bazie przeciwnika. W takiej sytuacji gracz atakujący dostaje punkt zwycięstwa. Punktów tych nie zdobywa się podczas, nawet zwycięskiej, obrony.

Podczas ataku obie strony ponoszą straty. Straty wylicza się według następującego schematu:

1. Zsumuj siłę ataku jednej ze stron (SA),
2. zsumuj zdolność obrony drugiej strony (SO)
3. ATAK UDANY - jeżeli SA – SO > 0, zbij wszystkie jednostki broniące, a straty atakującego to: X * (SA / SO), gdzie X to liczba jednostek danego typu.
4. ATAK NIE UDANY - w przeciwnym wypadku dla każdego typu jednostki policz ile jednostek straciła każda ze stron:
    a. broniący: X * (SA / SO), gdzie X to liczba jednostek danego typu.

b. atakujący: X * (SO / SA), gdzie X to liczba jednostek danego typu.

ARCHITEKTURA

Projekt oparty ma być o architekturę klient–serwer. Każdy z komponentów może składać się z jednej lub wielu aplikacji, każda tworzyć może jeden lub więcej procesów. Procesy nie mogą komunikować się ze sobą za pomocą mecha- nizmów niewyspecyfikowanych w poniższych opisach komponentów.

KLIENT

służy jako interfejs dla gracza:

- może być zrealizowany w trybie graficznym lub w (wygodnym) trybie tekstowym
- na bieżąco wyświetla aktualny stan gry
- nie przechowuje stanu gry lokalnie (pomijając dane aktualnie wyświetlane)
- nie wykonuje żadnego przetwarzania logiki gry (z wyjątkiem prostej obróbki danych przed wyświetleniem).
- *komenda „odśwież” — -0.5 oceny*

SERWER

- realizuje logikę gry
- obsługuje trzech klientów jednocześnie
- przechowuje stan gry
- w danej chwili obsługuje tylko jedną grę
- informuje klientów o zdarzeniach w grze, obsługuje ich komendy
- składa się z dwóch części:
    - części odpowiedzialnej za komunikację z klientami
    - części odpowiedzialnej za obsługę logiki oraz zdarzenia „asynchroniczne“ (budowa jednostek, przyrost surowców, walki)
- komponenty serwera komunikują się za pomocą pamięci współdzielonej z wykorzystaniem semaforów

ZALICZENIE

Wykonany samodzielnie projekt, skompresowany w jednym archiwum, należy przesłać na adres prowadzącego zajęcia w terminie przez niego wskazanym. Archiwum, nazwane imie.nazwisko.indeks.tar.gz, zawierać musi:

- pełne źródła aplikacji, kompilujące się bez ostrzeżeń (flaga -Wall kompilatora)
- skrypt do kompilacji lub plik Makefile
- plik tekstowy README zawierający:
  - instrukcję kompilacji
  - instrukcję uruchomienia
  - krótki opis zawartości poszczególnych plików *.c
- plik tekstowy PROTOCOL opisujący protokół komunikacji między komponentami projektu, w szczególności dokładny opis struktur przesyłanych kolejkami komunikatów oraz opis struktury pamięci współdzielonej.
- archiwum nie powinno zawierać zbędnych plików binarnych (produktów kompilacji).

Podstawą oceny jest terminowe oddanie projektu zgodnego z powyższą specyfikacją. Oddanie projektu z niepełną funkcjonalnością lub błędami skutkować będzie obniżeniem oceny. Oddanie projektu po terminie (w sesji poprawkowej) oznacza obniżenie oceny o 1. Oddanie po sesji poprawkowej skutkuje oceną **2.0 z laboratorium**.

Subiektywna ocena look&feel aplikacji (i jej kodu) może, lecz nie musi, wpłynąć dodatnio na ocenę projektu.

Wykrycie plagiatu skutkuje automatyczną oceną niedostateczną dla wszystkich zaangażowanych.