Projekt zaliczeniowy

Mateusz Hołenko Andrzej Stroiński Piotr Zierhoffer

21 grudnia 2015

Część I
Opis projektu

Celem projektu jest stworzenie dwuosobowej gry strategicznej, opartej o mechanizmy komunikacji między procesami systemu Linux, poznane na zajęciach laboratoryjnych z Systemów Operacyjnych.

Gra realizować ma zdefiniowane poniżej zasady rozgrywki oraz założenia architektoniczne. Ocenie nie podlega szata graficzna gry, a odpowiedznie wykorzystanie mechanizmów systemu Linux.

1 Zasady gry

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

1.1 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 ilość 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.

1.2 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.

1.3 Walka

Gracz w dowolnym momencie może zdecydować się na wydanie komendy ataku. W tym celu definiuje ilość 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.

1.3.1 Wyliczanie strat w jednostkach

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

1. zsumuj siłę ataku jednej ze stron (S_A),

	Atakujący A	Broniący B
Lekka piechota	2	7
Ciężka piechota	7	5
Robotnicy	3	3

Tablica 2: Rozkład sił walki przykładowej

2. zsumuj zdolność obrony drugiej strony (S_B),
3. jeżeli $S_A - S_B > 0$, wszystkie jednostki strony B giną,
4. w przeciwnym wypadku, dla każdego typu jednostki wylicz, ilu członków straciła: $\lfloor X * S_A/S_B \rfloor$, gdzie X to ilość jednostek danego typu.

Wyliczenia te należy powtórzyć dla drugiej strony, niezależnie od tego, kto atakował.

Przykład: Rozpatrzmy walkę na podstawie przedstawionego w tabeli 2 rozkładu sił.

1. Siła atakującego A: $2 * 1 + 7 * 1.5 + 3 * 3.5 = 23$
2. Zdolność obrony B: $7 * 1.2 + 5 * 3 + 3 * 1.2 = 27$
3. $S_A < S_B$, a więc należy wyliczyć straty broniącego:
   - lekka piechota — $\lfloor 7 * 23/27 \rfloor = \lfloor 5.963 \rfloor = 5$
   - ciężka piechota — $\lfloor 5 * 23/27 \rfloor = \lfloor 4.259 \rfloor = 4$
   - jazda — $\lfloor 3 * 23/27 \rfloor = \lfloor 2.556 \rfloor = 2$
   - strona B traci 5 jednostek lekkiej piechoty, 4 jednostki ciężkiej piechoty i 2 jednostki jazdy
4. Następnie należy wyznaczyć straty atakującego.
5. Siła broniącego B: $7 * 1 + 5 * 1.5 + 3 * 3.5 = 25$
6. Zdolność obrony atakującego A: $2 * 1.2 + 7 * 3 + 3 * 1.2 = 27$
7. $S_B < S_A$, więc należy wyliczyć straty atakującego:
   - lekka piechota — $\lfloor 2 * 25/27 \rfloor = \lfloor 1.852 \rfloor = 1$
   - ciężka piechota — $\lfloor 7 * 25/27 \rfloor = \lfloor 6.481 \rfloor = 6$
   - jazda — $\lfloor 3 * 25/27 \rfloor = \lfloor 2.778 \rfloor = 2$
   - strona A traci 1 jednostkę lekkiej piechoty, 6 jednostek ciężkiej piechoty i 2 jednostki jazdy

2 Ogólna 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 mechanizmów niewyspecyfikowanych w poniższych opisach komponentów.

2.1 Klient

- służy jako interface dla gracza
- może być zrealizowany w trybie graficznym lub w (wygodnym) trybie tekstowym
- komunikuje się z serwerem za pomocą kolejki komunikatów lub łączy nazwanych (-0.5 oceny)
- na bieżąco wyświetla aktualny stan gry
  - komenda „odśwież“ — -0.5 oceny (dotyczy pełnej aplikacji, patrz p. 2.4)
- nie przechowuje stanu gry lokalnie (pomijając dane aktualnie wyświetlane)
- nie wykonuje żadnego przetwarzania logiki gry (z wyjątkiem prostej obróbki danych przed wyświetleniem).

2.2 Serwer

- realizuje logikę gry
- obsługuje dwóch 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

2.3 Protokół

Format komunikatów wymienianych przez klientów i serwer jest dowolny. Protokół musi dopuszczać na przesłanie jednym komunikatem zlecenia ataku, informacji o jego rezultacie lub pełnego stanu gracza (lista wojsk + surowce).

2.4 Wersja uproszczona

Możliwe jest wykonanie uproszczonej wersji aplikacji, bez wykorzystania pamięci współdzielonej i semaforów.

W takiej sytuacji serwer zajmuje się tylko przekazywaniem komunikatów między klientami. Każdy klient zajmuje się obsługą własnej logiki, przyrostem surowców oraz zarządzaniem jednostkami. Obsługą walk zajmuje się klient broniący się — po upływie koniecznego czasu odsyła informację, za pośrednictwem serwera, do atakującego.

Wykonanie wersji uproszczonej obniża maksymalną możliwą ocenę za projekt o 1.

Część II
Forma zaliczenia

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 końcowej. Oddanie projektu po terminie oznacza obniżenie oceny o 0.5 za każdy rozpoczęty tydzień zwłoki.

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.