Przedmiot: ZPR - Zaawansowane Programowanie w C++
Zadanie: Heroes z AI
Zespół: TNT
Członkowie: Tomasz Nejman, Tomasz Naszkowski

Treść zadania:
Uproszczona wersja Heroes III. Nie oczekuję oprawy dźwiękowej, nie oczekuję animacji ani zawodowej oprawy graficznej, nie oczekuję wielu ras. Oczekuję grywalnej wersji gry człowieka z komputerem. Działa (w uproszczonej postaci) odkrywanie mapy, prowadzenie uproszczonych walk. Sprity można pobrać np. https://www.spriters-resource.com/pc_computer/heroes3/. Proszę skupić się na rozwoju AI w grze (walki) można użyć wielu technik, np. algorytmu https://pl.wikipedia.org/wiki/Algorytm_alfa-beta minimax z przycinaniem alfa-beta. Przed rozpoczęciem realizacji projektu proszę zapoznać się z zawartością http://staff.elka.pw.edu.pl/~rbiedrzy/ZPR/index.html.

✅❌
Lista Funkcjonalności:

1. Mapa gry:
   ✅ Które funkcjonalności udało się zrealizować:
   - Stworzenie Mapy, z dowolnością doboru typu pól (terenu)
   - Umieszczanie na mapie wrogich bohaterów wraz z ich armiami
   - Umieszczanie na mapie artefaktów.
   - Umieszczanie na mapie przeszkód.

   ❌Których nam się nie udało:
   - Tworzenie budynków, zamków, podziemi, dróg i logiki wody.

   a. Możliwość tworzenia swojej własnej mapy w tym umiejscowienie:
      i. Pól przechodnich (oraz ich terenu naturalnego, np. dla “Fortecy” -> śnieg i itp.),
      ii. Jednostek neutralnych,
      iii. Miast i ich właścicieli,
      iv. Terenu nieprzechodniego(np. Góry, lasy, bagna),
      v. Artefaktów,
      vi. Dróg,

vii. Pól wodnych,
viii. Ścian podziemi,
ix. Pustki podziemi.

b. Miasta:
   i. Wkraczanie bohaterów do miasta i garnizonowanie jednostek,
   ii. Budowanie budynków
   iii. Wykorzystywanie budynków:
      1. Rekrutacja jednostek
      2. Zakup i uzupełnianie księgi czarów
      3. Wymiana materiałów
      4. Rekrutacja bohaterów

2. Bohaterowie:

✅ Które funkcjonalności udało się zrealizować:

- Tworzenie bohaterów z atrybutami.
- Postacie mają możliwość poruszanie się po mapie.
- Podnoszenie artefaktów i zmiana statystyk z tym związana.
- Posiadanie armii, która może brać udział w walce.
- Bohaterzy mogą atakować innych bohaterów i jeśli wygrają to przeciwnik podlega dezintegracji.

❌ Których funkcjonalności nie udało nam się zrealizować:

- Bohaterzy nie mogą korzystać z magii.
- Ze względu na brak budynków do rekrutacji, nie ma możliwości rekrutacji jednostek.

a. Atrybuty (Atak, defensywę, moc umiejętności, mądrość),
b. Umiejętności (np. logistyka, ofensywa),
c. Artefakty używane,
d. Plecak z aktualnie nie używanymi artefaktami,
e. Czary i punkty magii (mana), niezbędne do rzucania czarów,
f. Armia. Składającą się z 7 stacków jednostek.
g. Możliwość poruszania się po mapie i wykonywania akcji takich jak:
   i. Rekrutacja jednostek z:
      1. Budynków rekrutacyjnych porozrzucanych po mapie,
      2. Siedlisk znajdujących się w miastach,

ii. Oznaczania budynków z zasobami i siedliskami,
iii. Walkę z innymi graczami lub jednostkami neutralnymi,
iv. Walkę w budynkach neutralnych tj. Smocza utopia,
v. Wymianę jednostek między bohaterami lub garnizonami w mieście,
vi. Wykorzystania punktów ruchu, by przemieszczać się.
vii. Rzucanie czarów pomocniczych tj. przywołanie łodzi.

3. Walka:

✅ Które funkcjonalności udało się zrealizować:

- W walce udało nam się zrobić pole bitwy, na którym mogą poruszać się jednostki jak i atakować inne jednostki.
- Rundy wykonują się w określonej kolejności, gdzie najszybsze jednostki mają możliwość najwcześniejszego ruchu.
- By ułatwić wizualnie walkę jest dodane podświetlanie aktualnie możliwych ruchów jakie dana jednostka może wykonać.
- Udało nam się zrobić trzy algorytmy sztucznej inteligencji:
  - MiniMax jednowątkowy z odcinaniem Alpha i Beta
  - MiniMax wielowątkowy z odcinaniem Alpha i Beta, które nie jest aż tak efektywne, ze względu na charakterystykę algorytmu.
- Nasze algorytmy Minimax opierają się na dwóch heurystykach:
  - Podstawowa: licząca różnicę w sile armii.
  - Dodatkowa: Wspomagająca atak silniejszej strony oraz ucieczkę słabszej strony.
- By zmniejszyć wielkość obiektu BattleField stworzyliśmy proxy co pozwoliło nam zaoszczędzić znaczną ilość miejsca, co było ważne ze względu na wykładniczą naturę problemu.

❌ Których funkcjonalności nie udało nam się zrealizować:

- Nie udało nam się stworzyć przeszkód, które by wzbogacały pole bitwy.
- Nie udało nam się zrobić magii.

a. Symulacja terenu heksagonalnego z nieprzekraczalnymi barierami

b. Symulacja przebiegu rund i kolejności wykonywania ruchów.
c. Wydawanie rozkazów jednostkom tj. idź, zaatakuj, broń się, czekaj.
d. Symulacja sztucznej inteligencji przeciwnika używając algorytmu alpha-beta i heurystyki biorącej pod uwagę maksymalizację różnicy mocy walczących armii, liczonych od statystyk poszczególnych jednostek.

4. Przeciwnicy:
   ✅ Które funkcjonalności udało się zrealizować:
   - Poziom trudności przeciwnika jest ustawiony arbitralnie w kodzie na 4, lecz można zmieniać.

   ❌ Których funkcjonalności nie udało nam się zrealizować:
   - Nie mamy zaimplementowanej logiki chodzenia przeciwnika na zewnątrz, co też oznacza brak archetypów.

   a. Archetypy przeciwników, np.:
      i. Odkrywca (próbuje odkryć jak najwięcej mapy)
      ii. Domyślny (zbalansowana strategia)
      iii. Generał (zbiera jak największą armię)
      iv. Budowlaniec (zależy mu na zabudowie swoich miast)
      v. Hazardzista (atakuje jeśli widzi szanse na powodzenie)
      vi. Bezpieczny (atakuje tylko jak wie, że wygra)
   b. Poziom trudności:
      i. Inne poziomy głębokości sprawdzania w algorytm alpha-beta.

5. Możliwość zapisu i odczytu stanu gry
   ✅Mamy szczątkową implementację zapisu i odczytu stanu gry.
   ❌Jest tylko szczątkowa.
6. Czytelny i działający GUI zintegrowany z backendem
   ✅Mamy działający GUI w trakcie bitwy, który umożliwia na poruszanie swojej armii oraz są pokazane wszystkie możliwe ruchy w danym momencie.
   ❌Nie posiadamy GUI obsługującego wyświetlanie danych danej jednostki.
7. Testy i analiza wydajności

✅Posiadamy w naszym programie testy jednostkowe, lecz ze względu na charakter projektu testowanie głównie odbywało się w domenie dynamicznej.

❌Nie posiadamy analizy wydajnościowej.

8. Ulepszanie heurystyk w algorytmie alpha-beta z użyciem RL, poprzez wielokrotne uruchamianie rozgrywki boty vs. boty:

❌Jak wcześniej już wspomnieliśmy, nie posiadamy implementacji sztucznej inteligencji typu RL.

   a. Stanami będą krotki czteroelementowe, składające się z:
      i. Zasoby (znormalizowane do złota przy wymianie “idealnej”, lecz zależnej od zamku, np. lochy potrzebują dużo siarki i mało kryształu),
      ii. Możliwości wytwarzania armii (moc armii jaką możemy wyprodukować z siedlisk posiadanych w ciągu tygodnia),
      iii. Armii (moc armii u bohaterów i w zamkach),
      iv. Względnej siły w porównaniu do wrogów (stosunek siły do siły przeciwników, gdzie siła przeciwników może być szacowana z gildii złodziei).
   b. Akcjami będą ruchy danego bota np.
      i. Atakuj
      ii. Uciekaj
      iii. Eksploruj
      iv. Eksploatuj

Czas pracy:

Nad projektem można uśrednić i stwierdzić, iż obaj razem spędziliśmy około 300 godzin na ten projekt i mamy zamiar kontynuować pracę nad nim.

Problemy napotkane:

Głównym problemem w naszym projekcie było niedostosowanie zakresu i złożoności projektu do dostępnych zasobów e.g. liczebności kadry. Problemem, który nas spotkał było zamaszyste podejście do projektu, co spowodowało, że duża część kodu, który napisaliśmy jest przydatna, lecz nie używana.

Tabela wylistowująca zadania i szacowany czas ich wykonania

ZADANIE		CZAS(h)	Faktyczny czas (h)
Stworzenie mapy, jej eksploracja i eksploatacja	Budowanie mapy	8	10
	Poruszanie się z wykorzystaniem algorytmu A*	2	2*
	Przejście między podziemiami i powierzchnią ziemi	0.5	N/A
	Ograniczenie widoczności dla każdego gracza (terra incognita)	0.5	N/A
	Interpretacja graficzna mapy i obiektów znajdujących się na niej w tym stworzenie minimapy	16	40
Stworzenie szablonu postaci i artefaktów	statystyki postaci	0,5	2
	umiejętności, wyposażenie, plecak	10	15
	artefakty i ich efekty	2,5	5
	Menu postaci	4	N/A
	czary	4	0,5*
Stworzenie symulacji rekrutacji i zarządzania armią	Możliwość powiększania swojej armii o jednostki rekrutowane w miastach	3	N/A
	Możliwość powiększenia swojej armii o jednostki rekrutowane na mapie (neutralne siedliska)	1	N/A
Symulacja sztucznej inteligencji	Symulacja logicznych ruchów przeciwników w czasie walki z wykorzystaniem algorytmu minimax z odcinaniem alpha-beta	10	40
	Symulacja logicznych ruchów przeciwnika na mapie z wykorzystanie algorytmu minimax z odcinaniem alpha-beta oraz uwzględnieniu różnych archetypów.	20	N/A
	Trenowanie hiperparametrów w heurystykach algorytmu minimax z użyciem RL, tj. uruchomienie wielu partii, z uczestnictwem tylko i	6	N/A

	wyłącznie botów, podczas których faworyzowane będzie nie tylko zwycięstwo, ale też zgodność stylu gry z archetypem postaci		
Stworzenie interaktywnego interfejsu graficznego	Mapa oraz minimapy(opcjonalne) + osobna mapa na czas walki	16 (wliczone już powyżej)	jak wyżej
	Menu bohatera	4 (wliczone już powyżej)	jak wyżej
	Zapis i wczytywanie gry	1	N/A
	Menu startowe	4	N/A
	Tabela wyników (gildia złodziei)	2	N/A
	Tabela bohaterów	1	N/A
	Tabela miast	1	N/A
Zapis i odczyt stanu gry		12	5
Złożenie wyżej wymienionych elementów w integralną i grywalną całość		10	15
	SUMA CZASU	118	

Zadania, które nie zostały wymienione w początkowym zakresie, a zajęły istotną część czasu:

- Generyczne tworzenie jednostek z frakcji:
  - 30 h
- Wczytywanie informacji użytkownika o ruchach, i.e. kliknięcie w odpowiednie pola na mapie, buforowane operacje I/O na mapie (WSAD):
  - 15 h
- Zbudowanie oraz możliwość poruszania i gry na polu bitwy w taki sposób by zajmowało mało pamięci. 20 h

Wykaz linii kodu:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                             37            471            306           2839
C/C++ Header                    69            430            443           1808
```

CMake	25	109	97	625
YAML	2	7	2	274
JSON	5	0	0	185
Markdown	5	14	0	113
make	2	47	45	98
Bourne Shell	1	0	0	8
TypeScript	1	0	0	2
SUM:	147	1078	893	5952