Heroes of Might and Magic 3: Bot – opis rozwoju projektu

Sprawozdanie z przedmiotu

Zintegrowane Systemy Decyzyjne 2

Grupa HoMM3 Bot

Grudzień 2022

1. Wstęp

Celem projektu grupowego w ramach przedmiotu ZSD 2, było rozwijanie funkcjonalności bota prowadzącego rozgrywkę w grze Heroes of Might and Magic 3 Hota HD, stworzonego w ramach przedmiotu ZSD. Główną funkcjonalnością, która została rozwinięta, jest możliwość grania na dowolnej mapie[1]. Dodatkowo rozwinięto i zoptymalizowano mechaniki wykorzystywane zarówno podczas rozgrywki na mapie przygody jak i w bitwie.

W pewnym stopniu ten raport można traktować jako rozwinięcie raportu przedstawionego w czerwcu 2022 roku w ramach pierwszej części przedmiotu. Technologie takie jak:

- Tesseract
- CNN
- QNN
- Algorytmy hierarchiczne

Zostały przedstawione w raporcie z czerwca, stąd nie będą tłumaczone w tym dokumencie. Zamiast tego zostanie skupione się na nowych rozwiązaniach, które zostały zrealizowane w ramach rozwoju projektu homm3 bot.

2. Przegląd podobnych rozwiązań

2. 1 VCMI

VCMI jest silnikiem typu open-source dla gry Heroes of Might and Magic III napisanym w języku C++. Celem projektu VCMI jest przepisanie całego środowiska HOMM 3 od podstaw, dając mu nowe i rozszerzone możliwości. Projekt aktualnie nie jest ukończony, wciąż wymaga wielu elementów i funkcji do pełnej funkcjonalności.

Udostępniony zestaw deweloperski do rozwoju VCMI jest słabo udokumentowany, ponadto jego niektóre funkcje przeszkadzałyby naszemu zespołowi w rozwoju własnego bota. Z tego powodu zdecydowaliśmy się porzucić pomysł zastosowania wymienionego rozwiązania.

2. 2. The Application of Co-Evolutionary Genetic Programming and TD Reinforcement Learning in Large-Scale Strategy Game VCMI (Wilisowski, Dreżewski)

[1] Bez wody, podziemii, teleportów a wielkość wynosi 72x72.

Artykuł proponujący rozwiązanie bitwy poprzez m.in. uczenie ze wzmocnieniem oraz zastosowanie projektu VCMI, który zapewniał środowisko testowe poprzez wprowadzenie modyfikowalnego agenta do bitwy. Ciekawym elementem pracy było stworzenie systemu ewaluacji ruchów na podstawie estymowanych wyników akcji, którymi były na przykład: procent jednostek zasięgowych chronionych czy liczba pokonanych przeciwników. W podobny sposób my zrealizowaliśmy bitwę w naszym projekcie.

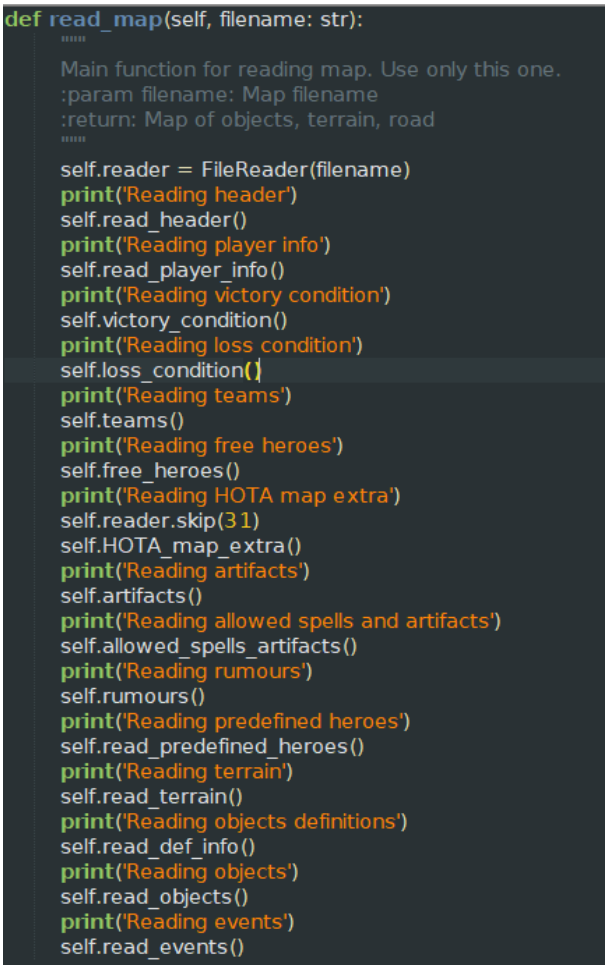
2. 3. https://github.com/TheDude-gh/heroes3mapscan

Użytkownik ‘TheDude-gh’ stworzył parser map w języku PHP, który dekoduje pliki h3, za pomocą selekcji bajtów odpowiadającym cechom map/różnym obiektom na mapie. W rozwiązaniu problemu dekodowania map, zostało zainspirowane się tym systemem.

3. **Rozwiązanie**

3. 1 Czytanie dowolnej mapy.

Czytanie mapy odbywa się poprzez czytanie grup bajtów z plików h3m. Określone grupy bajtów zawierają informację o różnych elementach mapy. Przykładowo, grupa pierwszych 30 bajtów to nagłówek, który zawiera informacje o wersji mapy, podwersji określonej przez dodatek HoTA HD, wielkość mapy itd. Kolejne grupy zawierają informację o bardzo wielu elementach (artefakty, warunki zwycięstwa, …), jednak największą grupą jest ta, która reprezentuje teren. Ta grupa może zawierać dziesiątki tysięcy bajtów zależnie od ilości obiektów na mapie. Wiedza o grupach bajtów i ich długości czerpana została zarówno z silnika z VCMI oraz źródła przedstawionego 2 punkcie **2.3.** W programie parser został napisany w języku python na podstawie tych dwóch źródeł. Na rys. 1 pokazany został fragment kodu, którego celem jest przeczytanie mapy przez wywoływanie kolejnych funkcji biorących informację o danym elemencie mapy.



```
def read_map(self, filename: str):
    """
    Main function for reading map. Use only this one.
    :param filename: Map filename
    :return: Map of objects, terrain, road
    """
    self.reader = FileReader(filename)
    print('Reading header')
    self.read_header()
    print('Reading player info')
    self.read_player_info()
    print('Reading victory condition')
    self.victory_condition()
    print('Reading loss condition')
    self.loss_condition()
    print('Reading teams')
    self.teams()
    print('Reading free heroes')
    self.free_heroes()
    print('Reading HOTA map extra')
    self.reader.skip(31)
    self.HOTA_map_extra()
    print('Reading artifacts')
    self.artifacts()
    print('Reading allowed spells and artifacts')
    self.allowed_spells_artifacts()
    print('Reading rumours')
    self.rumours()
    print('Reading predefined heroes')
    self.read_predefined_heroes()
    print('Reading terrain')
    self.read_terrain()
    print('Reading objects definitions')
    self.read_def_info()
    print('Reading objects')
    self.read_objects()
    print('Reading events')
    self.read_events()
```

Rys 1. Fragment kodu przedstawiający sposób czytania mapy

3. 2 Rozwijanie adventure AI.

Rozwój adventure AI polegał w dużej mierze na naprawę błędów, które znacznie zmniejszały jakość rozgrywki bota. Rozwiązywanie ich zajęło dużą ilość czasu zespołowi zajmującego się tą częścią projektu. Niemniej, zostały dodane też nowe funkcjonalności, które swoją obecnością zwiększają możliwości bota. Tymi dodatkami są:

- Adaptacyjne dobieranie liczby bohaterów
- Wymiana jednostek między bohaterami

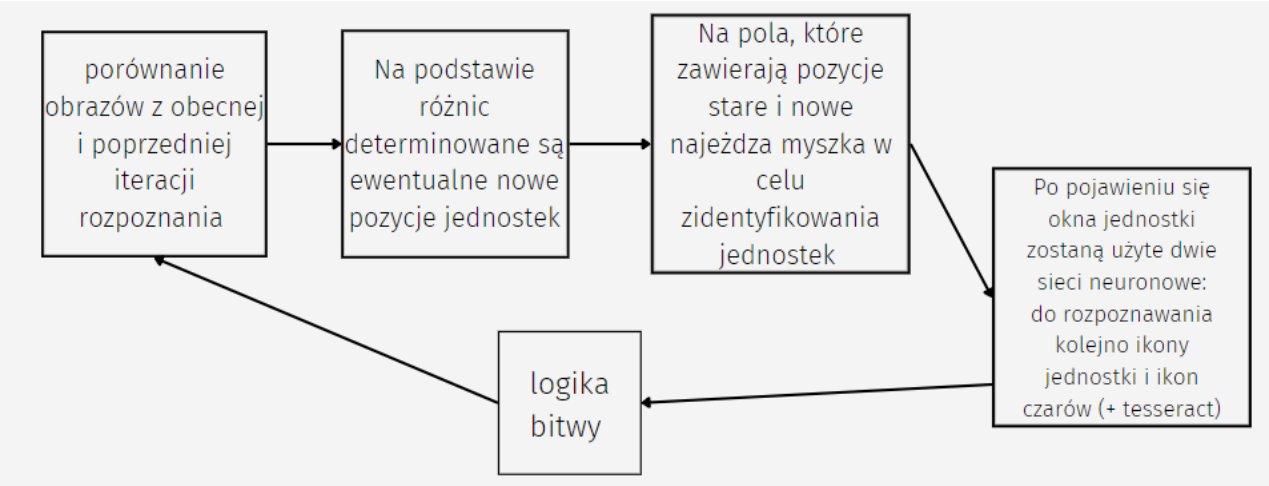
Pochylono się również nad samym obliczaniem wartości w funkcjach hierarchicznych, które zostało ulepszone podczas licznych testów agenta.

Skończono również nowy system wykrywania pozycji bohatera i innych obiektów po kursorze, co znacznie przyspiesza agenta, lecz niestety nie działa on u większości osób w grupie, przez co nie został finalnie zastosowany (choć jest nadal w repozytorium zapisany).

Założenia projektowe dotyczące tej części zostały spełnione, gdyż zakładana była szeroko pojęta optymalizacja działania adventure AI i rzeczywiście, została spełniona.

3. 3 Rozwijanie battle AI.

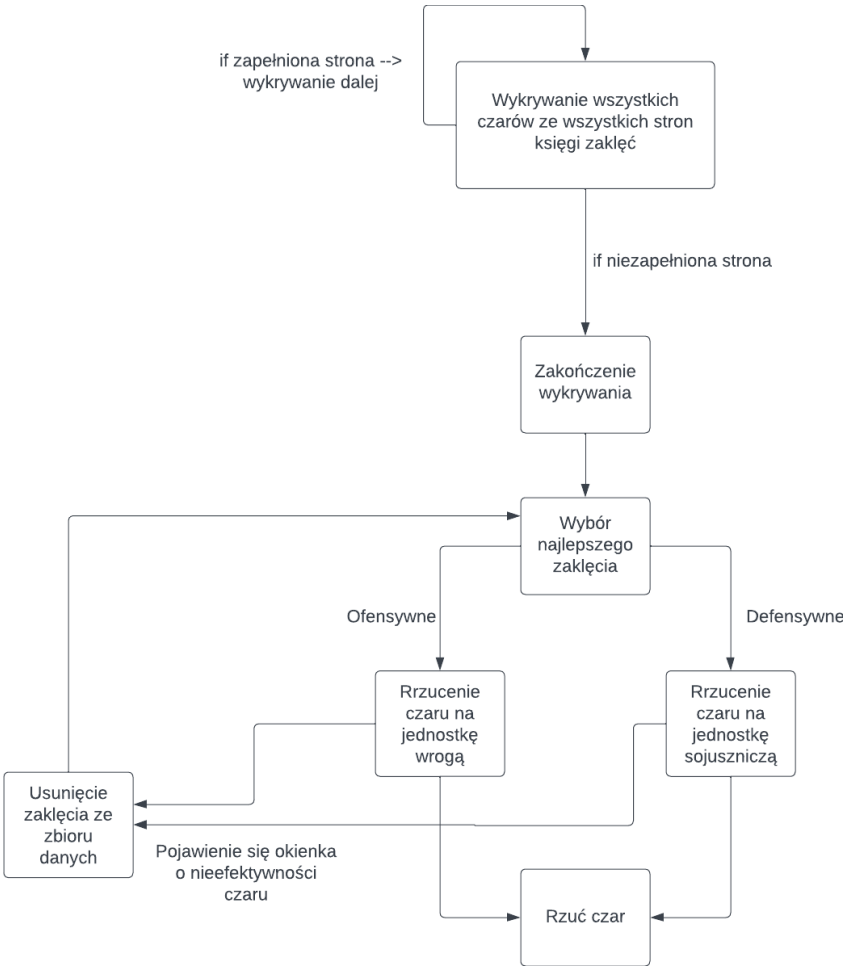
Założenia projektowe odnośnie battle AI mówiły o włączeniu do bitwy machin wojennych i ulepszonego systemu rzucania zaklęć. Obie te rzeczy zostały zrealizowane, przy czym schemat blokowy przedstawiający działanie nowego systemu został pokazany na rys 4. Dodatkowo, został stworzony nowy system rozpoznawania jednostek w bitwie, który został przedstawiony na rysunku 2.



Rys 2. Schemat blokowy nowego rozpoznawania jednostek w bitwie.

W odróżnieniu od poprzedniego systemu, polegałby na różnicach obrazów w celu “przypuszczenia” gdzie znajdują się ewentualne jednostki, a następnie potwierdzałby to faktem pojawienia się karty jednostki (pokazanej na rys 3., która przechowuje ikonę na podstawie której, można rozpoznać jaka to jednostka oraz ikony czarów, które dotykają tej jednostki. Używane są dwie sieci neuronowe (dla ikony jednostki i dla ikony czarów) w celu rozpoznania konkretnej jednostki i to jakie czary jej dotyczą. Dodatkowo używany jest tesseract do odczytywaniu tekstu na karcie, który bezpośrednio dotyczy statystyk jednostki.

Rys 3. Karta jednostki



Rys 4. Schemat blokowy przedstawiający nowy system magii.

3. 4 Rozwijanie innych elementów systemu.

Najważniejszym elementem poza odczytem map oraz stałymi, już znanymi częściami projektu (adventureAI, battleAI), była realizacja autozapisu gry oraz możliwość wczytania jej w dowolnym momencie. Polegała ona na przechowywaniu wszystkich obiektów i zmiennych odnoszących się do danego momentu gry. Te dane są ściśle związane z danym autozapisem, który występuje co turę.

4. **Wyniki**

4. 1 Czytanie mapy

Zdolność odczytywania map została oceniona przez wczytanie 28 plików typu h3m zawierających mapy. Z powodzeniem zdekodowanych zostało 26 map (**~93%**), co jest potwierdzeniem skuteczności systemu. Dwie mapy, które nie zostały przeczytane, nie mogą również zostać przeczytane przez system przedstawiony w podpunkcie **2.3**, co może sugerować korupcję mapy. Agent grający na zdekodowanych mapach zachowuje wszelkie zdolności, które przedstawiał na mapie “zhardkodowanej”.

4. 2 BattleAI

Wraz z dodaniem nowych funkcjonalności, przyszły nowe wyniki. Nowy system rozpoznawania spowodował, że błędy odczytu zostały praktycznie zniwelowane do zera. Czas trwania tury zależy od tego ile jednostek będzie na polu bitwy. Przy małej liczbie jednostek jest on znacznie mniejszy niż w poprzednim systemie. Przy skrajnie dużej ilości jednostek na polu bitwy, jest on trochę dłuższy. Średnio odnotowano przyspieszenie czasu trwania tury o ok. **1,5** [s] w porównaniu do poprzedniego systemu. Przyspieszenie jednak nie jest tutaj największym priorytetem, a sama precyzja tego sposobu, która zniwelowała błędy odczytu. Czas rzucania czarów wydłużył się (choć nieznacznie), ze względu na to, że obecny system obsługujący tą czynność jest zwyczajnie bardziej skomplikowany niż wcześniejszy. Biorąc pod uwagę jego zalety, ewentualne wydłużenie operacji rzutu zaklęcia, jest nieważne.

Sam wskaźnik zwycięstw wzrósł w porównaniu z tym, znajdującym się w czerwcowym raporcie. Agent wygrywa więcej walk, zakładając, że używa czarów. Nowy system zaklęć jest bardzo intuicyjny i często rzucane są czary masowe, które są niezwykle silne.

Wygrane	Przegrane	Ilość gier	Skuteczność
19	11	30	63%

Tab. 1 – statystyki zagranych walk w battleAI

4. 2 AdventureAI i inne

Funkcjonalności, które ulepszają działanie systemu takie jak adaptacyjne dobieranie liczby bohaterów, zmiany systemu wartości, czy wymiana jednostek powodują, że system jest potężniejszy i działa bardziej “jak człowiek”. Usunięto wiele błędów, które powodowały zacięcie się programu. Finalnie stworzyliśmy system, który przyspiesza rozgrywkę za pomocą detekcji kursora, jednak nie działa on u większości osób w projekcie (podejrzenie pada na ustawienia systemowe), stąd nie znajduje się w stabilnej wersji programu.

Stworzony został nowy system autozapisu, który pozwala na zapisywanie i wczytywanie stanu gry w dowolnym momencie. Ułatwia to zdecydowanie pracę nad programem oraz pozwala na kontynuowanie rozgrywki w dowolnym momencie. Jest to zmiana “quality of life”.

5. Podsumowanie

Wszystkie założenia zostały spełnione. Udało się rozwinąć projekt w trzech różnych kierunkach z czego jeden był krokiem naprzód w nieznanym kierunku. Konkretniej, mowa jest tutaj o graniu na kompletnie dowolnej mapie za pomocą dekodowania plików h3 za pomocą narzędzia napisanego w języku python zrobionego przez grupę projektową. Był to najważniejszy cel i udało się go osiągnąć. Równocześnie ulepszyliśmy nasz dotychczasowy system, przez rozwój adventure AI i battle AI opisany w punktach **3.2** i **3.3.** Inne zmiany warte większej uwagi to autozapis, który pozwala na wczytywanie i zapisywanie stanu gry. Jest to bardzo przydatna zmiana zarówno pod względem testowo-programistycznym jak i jakością obsługi bota. W ten sposób można w dowolnym momencie wczytać agenta, który zacznie grać w momencie w którym skończył

6. Możliwości rozwoju

Mimo ekstensywnej pracy nad programem, ulepszeniem i dodaniu wielu elementów, nadal istnieje duże pole do zmian, szczególnie w adventure AI. W tym semestrze skupione zostało się na możliwości grania na dowolnej mapie, a battleAI i adventureAI zeszły na drugi plan. Kontynuacja projektu polegałaby na wyłącznym skupieniu się na jednym z dwóch tych elementów w celu stworzenia bardzo silnego logicznie oprogramowania.