|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | |  | **Wydział Informatyki i Zarządzania** kierunek studiów: Informatyka Praca dyplomowa - inżynierska**Implementacja algorytmów sztucznej inteligencji** **w grze Starcraft: Brood War**Marcin Żerko słowa kluczowe:  Neuroewolucja, algorytm genetyczny, sieć neuronowa  Strategia czasu rzeczywistego, gry komputerowe  Starcraft: Brood War  krótkie streszczenie:  Celem pracy jest implementacja algorytmów sztucznej inteligencji w grze Starcraft: Brood War. Zakres obejmuje przegląd możliwych technologii do wykorzystania w projekcie, ich implementacje, oraz badania ich skuteczności.   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | opiekun pracy  dyplomowej | .................................................. | | ....................... | | ....................... | | | *Tytuł/stopień naukowy/imię i nazwisko* | | *ocena* | | *podpis* | | | Ostateczna ocena za pracę dyplomową | | | | | | | | | Przewodniczący Komisji egzaminu dyplomowego | | | ..................................................  *Tytuł/stopień naukowy/imię i nazwisko* | | ....................... | | ....................... | | *ocena* | | *podpis* |   *Do celów archiwalnych pracę dyplomową zakwalifikowano do:\**   1. *kategorii A (akta wieczyste)* 2. *kategorii BE 50 (po 50 latach podlegające ekspertyzie)*   *\* niepotrzebne skreślić*   |  | | --- | | pieczątka wydziałowa | |
|  |  |  | Wrocław 2017 |

**Streszczenie**

To do

**Abstract**

To do

**Spis treści**

[1. Wstęp 7](#_Toc497594955)

[1.1. Wprowadzenie 7](#_Toc497594956)

[1.2. Geneza pracy 7](#_Toc497594957)

[1.3. Cel i zakres pracy 7](#_Toc497594958)

[2. Stan wiedzy i techniki w zakresie tematyki pracy 8](#_Toc497594959)

[2.1. Porównanie Starcrafta do innych gier 8](#_Toc497594960)

[2.2. Przegląd istniejących technologii 8](#_Toc497594961)

[2.3. Przegląd istniejących technik 9](#_Toc497594962)

[3. Założenia projektowe 10](#_Toc497594963)

[4. Implementacja aplikacji 11](#_Toc497594964)

[5. Badania skuteczności 12](#_Toc497594965)

[6. Podsumowanie 13](#_Toc497594966)

1. Wstęp

1.1. Wprowadzenie

Starcraft jest jedną z najpopularniejszych strategicznych gier czasu rzeczywistego. Gra została wydana w 1998 roku na komputery osobiste przez firmę Blizzard Entertainment. Rok później wydany został dodatek do niej, o podtytule Starcraft: Brood War. Rozpowszechnił się on na tyle, że niemal zawsze jest dodawany do podstawowej gry i tą wersją zajmę się w tej pracy. Gra odniosła ogromny sukces głównie w Korei Południowej i stworzyła się wokół niej duża społeczność esportowa. Celem gry jest zniszczenie wszystkich budynków przeciwników. Bardzo często gracze poddają się wcześniej, gdy uznają, że w ich sytuacji wygrana jest już niemożliwa. Aby to osiągnąć należy budować własne struktury, rozwijać w nich różne pomocne technologie, trenować jednostki bojowe i tak nimi pokierować, aby wyeliminować obronę przeciwnika. Wszystko to kosztuje – walutą w grze są kryształy minerałów, oraz rzadszy, ale potrzebny do bardziej zaawansowanych operacji, gaz – wespan. Możemy je wydobywać przy pomocy naszych jednostek zbierających, które produkujemy przy pomocy naszego głównego budynku. W grze są dostępne trzy rasy, które różnią się sposobem rozgrywki, oraz wyglądem.

1.2. Geneza pracy

Starcraft jest grą wieloosobową i może być rozgrywany przeciwko innym ludzkim przeciwnikom. Jednak nie zawsze jest taka możliwość, lub gracz może nie mieć na to ochoty. Wtedy na pomoc przychodzą boty, – czyli zaprojektowana sztuczna inteligencja, która próbuje udawać grę prawdziwego człowieka. Ta dołączona do instalacji gry jest prymitywna i łatwo z nią wygrać. W internecie jest dostępnych wiele innych botów, ale większość z nich skupia się wyłącznie na modyfikacji wysoko poziomej strategii, podchodząc bardzo podstawowo do kwestii taktycznego poruszania się jednostek. Przykładowo dzięki wycofywaniu pojedynczych rannych jednostek za linię stworzoną z pozostałych, jest możliwość uniknięcia jej śmierci – może ona wtedy wciąż zadawać pełne obrażenia przeciwnikom, będąc chroniona przez swoje jednostki sojusznicze, które przejmą na siebie ostrzał wroga i pozwolą jej przeżyć. W związku z tym postanowiłem połączyć dwie dziedziny swoich zainteresowań – sztuczną inteligencję oraz Starcrafta – i spróbować wypracować własne rozwiązanie, które pozwoliłoby na skuteczne zarządzanie tymi to jednostkami w czasie rzeczywistym, podczas gry.

1.3. Cel i zakres pracy

Celem pracy jest implementacja algorytmów sztucznej inteligencji w grze Starcraft: Brood War do zarządzania jednostkami podczas starcia. Zakres obejmuje przegląd możliwych technologii do wykorzystania w projekcie, ich implementacje, oraz badania ich skuteczności. W drugim rozdziale zostały opisane możliwe do wykorzystania technologie i podejścia, wraz z analizą istniejących rozwiązań. Rozdział trzeci zawiera informacje dotyczące projektu aplikacji, jego wymagań i przypadków użycia. W kolejnym rozdziale opisany jest sposób implementacji programu, diagram klas, oraz zostały zamieszczone widoki z aplikacji. Następny rozdział składa się z definicji miary skuteczności i wyników badań zastosowanych algorytmów. Na koniec opisane zostało podsumowanie projektu, wraz z dalszymi możliwościami rozwoju.

Należy jeszcze to zrewidować z faktycznym stanem rozdziałów w przyszłości. Wyszło ich pierwotnie 6.

2. Stan wiedzy i techniki w zakresie tematyki pracy

2.1. Porównanie Starcrafta do innych gier

W porównaniu do gier takich jak szachy, trudnością tej gry dla algorytmów sztucznej inteligencji są dwa fakty. Po pierwsze - jest ona rozgrywana w czasie rzeczywistym, a nie turowym. Powoduje to, że, czas na obliczenia jest znacznie ograniczony. Program musi odpowiadać na ruchy przeciwnika jak najszybciej, gdyż każde opóźnienie może skutkować utratą jednostek w starciu, lub niedopasowaniem strategii do odpowiedzi przeciwnika, co rezultacie zmniejsza szanse na wygraną w grze. Zaproponowane przez mnie rozwiązanie tego problemu zostanie przedstawione w dalszym rozdziale.

Dodatkowo też jest to gra niepełnej informacji. Gracz nie ma pełnej wiedzy o stanie i posunięciach przeciwnika, do póki nie widzi ich w danym momencie własnymi jednostkami lub budynkami. W związku z tym bot musi, albo losować strategie, albo wybierać je z pewnym prawdopodobieństwem. Gdy już uda się zwiadowcom dowiedzieć się o posunięciach przeciwnika, może wystąpić konieczność odpowiedniej modyfikacji naszych zachowań, aby nie zostać pokonanym. W pracy tej zajmuje się wyłącznie samym zarządzaniem jednostkami w starciu, gdy już wiemy, jakie siły posiada przeciwnik, w związku, z czym potencjalne możliwości na obejście tej kwestii nie zostaną tutaj przedstawione.

2.2. Przegląd istniejących technologii

Podstawową rzeczą, jaką trzeba wybrać przed implementacją programu, jest język programowania, jaki chcemy do tego celu użyć. Wiele projektów związanych ze sztuczną inteligencją używa do tego celu C++ - w to także te oparte na Starcrafcie. Jest to język ogólnego przeznaczenia, który między innymi dzięki możliwości bezpośredniego zarządzania pamięcią pozwala na bardzo efektywne wykorzystywanie zasobów sprzętowych. W związku z tym wymaga on od programisty więcej uwagi i ostrożności podczas implementacji, by uniknąć wycieków pamięci. Niestety moje umiejętności pisania w tym języku są zbyt małe, przez co zdecydowałem się używać do tego projektu bardziej znanego mi języka Java. Jest to język działający na własnej maszynie wirtualnej - dzięki temu jest bardziej przenośny, co pozwoli mi w łatwy sposób uruchamiać aplikację zarówno na komputerach z systemem operacyjnym Windows, jaki i Linux. Posiada on też automatyczny garbage collector, który zarządza pamięcią, pozbywając się nieużywanych obiektów, co przekłada się na łatwiejszą obsługę dla programisty. Wadą tego rozwiązania jest niemożliwość ręcznego zarządzania pamięcią, przez co jest on jednocześnie wolniejszy od swojego konkurenta. Uznałem jednak, że potencjalne zalety C++ mogą zostać przeważone przez ewentualne błędy w mojej implementacji, ze względu na brak mojego doświadczenia w tym języku w porównaniu do Javy.

Głównym źródłem informacji na temat zastosowania metod sztucznej inteligencji w Starcrafcie jest Starcraft AI Wiki [1]. Zawiera ona opis najpopularniejszych bibliotek, linki do dalszych stron, jak i wiele przydatnych porad dla osób próbujących swoich sił w tej dziedzinie. Najbardziej sprawiedliwym w stosunku dla ludzkich przeciwników naszego bota, było by odczytywanie ekranu gry, analizowanie tego, co widzimy i odpowiednie wykonywania akcji, tak żeby udawać manualne użycie klawiatury i myszki. Było by to jednocześnie dość trudne i wprowadzało kolejne poziomy abstrakcji. Zamiast tego lepszym rozwiązaniem wydaje się użycie odpowiedniej biblioteki, która udostępniłaby nam interfejs do otrzymywania danych z gry i wydawania rozkazów, dzięki czemu moglibyśmy pominąć ten cały problem. Jedynym kompletnym frameworkiem, który by nam na to pozwolił jest BWAPI [2]. Aby zachować sprawiedliwość domyślnie nie są udostępniane żadne wiadomości na temat przeciwnika, których bot nie mógłby zobaczyć grając w normalny sposób. Są także blokowane potencjalne rozkazy wydawane manualnie, aby człowiek nie mógł w żaden sposób pomagać sztucznej inteligencji. BWAPI nie wspiera najnowszej poprawki do gry – używa wersji 1.16.1 – rozwój Starcrafta był przez długi czas zawieszony i dopiero w tym roku został wznowiony. Aby dostosować się do zmian, musiałyby zostać zmienione konkretne offsety pamięci gry, z których korzysta do odczytywania i wydawania rozkazów. Było by to mocno pracochłonne, a nowe wersje gry nie wprowadzają do niej istotnych z tego punktu widzenia poprawek.

BWAPI jest napisane w C++, dlatego musiałem poszukać odpowiednego rozwiązania, które pozwoliłoby mi użyć go od strony Javy. Istnieje kilka technologii pozwalających złączyć ze sobą te dwa języki programowania, jednak zazwyczaj do własnej implementacji wymagają pewnej znajomości tego pierwszego języka programowania. Najbardziej aktualnym takim gotowym rozwiązaniem jest BWMirror [3]. Używanym w nim sposobem jest Java Native Interface [4]. Pozwala on na wykonywanie metod napisanych w obydwóch językach i przekazywanie pomiędzy nimi obiektów. Cechą tego portu jest to, że dane dotyczących parametrów jednostek i budynków wczytywana z gry jest przechowywana po stronie kodu w części C++. W związku z tym nie jest możliwe stworzenie tylko i wyłącznie za pomocą Javy symulatora, który pozwoliłby trenować sztuczną inteligencję bez uruchamiania gry. Inaczej jest w przypadku JNIBWAPI [5] – tutaj jest możliwe rozszerzenie biblioteki aby dane te po uruchomieniu gry były zapamiętane przez kod Javowy do plików i następnie wczytywane w razie potrzeby, bez konieczności ponownego uruchamiana gry. W przeciwieństwie do BWMirror, nie wspiera ona najnowszej, czwartej wersji BWAPI, pozostając na wersji trzeciej.

Sparcraft/Jarcraft/własna bilioteka

2.3. Przegląd istniejących technik

Genetyczne, sieci, neuroewolucja

3. Założenia projektowe

4. Implementacja aplikacji

5. Badania skuteczności

6. Podsumowanie

# Bibliografia

|  |  |
| --- | --- |
| [1] | „StarCraft AI,” [Online]. Available: http://www.starcraftai.com. [Data uzyskania dostępu: 04 10 2017]. |
| [2] | „BWAPI,” [Online]. Available: https://bwapi.github.io/. [Data uzyskania dostępu: 04 10 2017]. |
| [3] | „BWMirror API,” [Online]. Available: http://bwmirror.jurenka.sk/. [Data uzyskania dostępu: 04 10 2017]. |
| [4] | „Java Native Interface,” Oracle, [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/jni/. [Data uzyskania dostępu: 04 10 2017]. |
| [5] | „JNIBWAPI,” [Online]. Available: https://github.com/JNIBWAPI/JNIBWAPI. [Data uzyskania dostępu: 04 10 2017]. |