



**AGH**

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE  
Wydział Fizyki i Informatyki Stosowanej

---

---

## **Praca magisterska**

**Rafał Jabłonowski**

**Michał Grygiel**

kierunek studiów: **Informatyka Stosowana**  
specjalność **Informatyka w Nauce i Technice**

**AI w nowoczesnej grafice komputerowej na  
przykładzie inteligentnego i autonomicznego  
robota w grze komputerowej**

Opiekun: **dr hab. inż. Tadeusz Szuba; prof. nzw. AGH**

Kraków, luty 2012

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....  
(czytelny podpis)

.....  
(czytelny podpis)

Kraków, luty 2012

**Tematyka pracy magisterskiej i praktyki dyplomowej  
Rafała Jabłonowskiego i Michała Grygiela, studentów V roku studiów kierunku  
Informatyka Stosowana, specjalności Informatyka w Nauce i Technice.**

**Temat pracy magisterskiej: AI w nowoczesnej grafice komputerowej na  
przykładzie inteligentnego i autonomicznego robota w grze komputerowej**

Opiekun pracy: dr hab. inż. Tadeusz Szuba; prof. nzw. AGH

Recenzent pracy:

Miejsce praktyki podyplomowej: WFiIS AGH w Krakowie

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
  - idea Sztucznej Inteligencji,
  - zapoznanie się z Microsoft XNA Game Studio,
  - projektowanie gry oraz wstępnego modelu Sztucznej Inteligencji,
  - przygotowanie oprogramowania,
  - sporządzenie sprawozdania z praktyk.
4. Kontynuacja rozwoju fizyki gry, cząsteczek oraz modeli.
5. Rozbudowa istniejącego modelu Sztucznej Inteligencji.
6. Analiza i zatwierdzenie osiągniętych rezultatów przez opiekuna.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie: luty 2012

.....  
(podpis kierownika katedry)

.....  
(podpis opiekuna)

**Merytoryczna ocena pracy przez opiekuna:**

**Merytoryczna ocena pracy przez recenzenta:**

# **Abstract**

Artificial Intelligence in computer games area is in constant need of improvement to meet nowadays demands of the players. The aim of our thesis was to design and develop an intelligent, autonomic robot in a computer game based on intelligent agent and fuzzy logic. The work covers theoretical and practical issues. In the first part, we describe the theory of Artificial Intelligence with examples of using it in games and general fundamentals of modern computer graphics design. In the second part we present in detail how we designed and developed the game with implementation of intelligent agents and fuzzy logic behavior model.

---

Sztuczna Inteligencja w obszarze gier komputerowych jest w stałej potrzebie udoskonalania, aby sprostać współczesnym wymaganiom graczy. Celem naszej pracy było zaprojektowanie i rozwój autonomicznego robota w grze komputerowej, jako inteligentnego agenta wykorzystującego logiki rozmytej w swoim procesie decyzyjnym. Praca obejmuje zagadnienia teoretyczne jak i praktyczne. W pierwszej części opisujemy teorię Sztucznej Inteligencji wraz z przykładami jej zastosowania w grach oraz fundamentalne podstawy tworzenia nowoczesnej grafiki komputerowej. W drugiej części prezentujemy szczegółowo jak zaprojektowaliśmy i rozwinęliśmy grę z implementacją inteligentnego agenta wraz z modelem zachowawczym robota w logice rozmytej.

# Spis treści

Spis treści .....	8
1. Wprowadzenie.....	10
1.1 Motywacja.....	12
1.2 Cel pracy .....	12
1.3 Zakres pracy.....	13
2. Sztuczna Inteligencja .....	15
2.1 Czym jest Sztuczna Inteligencja.....	15
2.2 Rozwój Sztucznej Inteligencji na przełomie lat .....	17
2.3 Inteligentny agent .....	18
2.4 Logika rozmyta .....	26
2.5 Podstawowe definicje i właściwości Zbiorów Rozmytych .....	27
3. Sztuczna Inteligencja w grach.....	41
3.1 Krótka Historia.....	42
3.2 Inteligentny agent bazujący na logice rozmytej.....	43
3.3 Przykłady zastosowań AI w grach.....	48
4. Nowoczesna grafika komputerowa .....	58
4.1 Modele i geometria.....	59
4.2 Interaktywne przetwarzanie obrazu w czasie rzeczywistym .....	60
4.3 Odwzorowanie fizyki w nowoczesnej grafice komputerowej.....	70
5. Koncepcja inteligentnego i autonomicznego robota .....	73
5.1 Założenia projektowe gry .....	73
5.2 Koncepcja nowoczesnej grafiki komputerowej w grze .....	74
5.3 Koncepcja modelu Sztucznej Inteligencji .....	81
5.4 Rozmyte systemy wnioskujące.....	90
6. Realizacja .....	100
6.1 Nowoczesna grafika komputerowa w grze „ <i>Tank Wars</i> ” .....	100
6.2 Model Sztucznej Inteligencji.....	109

6.3 Miara skuteczności inteligentnego agenta .....	121
7. Podsumowanie i wnioski .....	127
Załączniki.....	129
Bibliografia .....	130
Spis ilustracji.....	134
Podział pracy .....	138
Zawartość DVD.....	139

*Chcemy podziękować  
Panu dr hab. inż. Tadeuszowi Szubie; prof. nzw. AGH  
za konsultacje, cenne wskazówki, rady oraz miłą atmosferę  
w trakcie pisania tej pracy.*

# 1. Wprowadzenie

„Jeśli maszyna zachowuje się tak inteligentnie jak człowiek, to jest ona tak inteligentna jak człowiek.” [1]

Alan Turing

Nazywamy siebie *Homo sapiens* – człowiek rozumny<sup>1</sup>, gdyż zdolność myślenia jest tak ważną dla nas cechą. Ludzkość przez setki lat próbuje zrozumieć w jaki sposób myślimy, to jest jak postrzegamy, rozumiemy, wnioskujemy i przewidujemy w świecie wiele większym i bardziej skomplikowanym od nas samych. W swojej renomowanej książce „*Artificial Intelligence: A Modern Approach*” [3], panowie Russell i Norvig przedstawiają Sztuczną Inteligencję jako dziedzinę idącą dużo dalej: nie próbuje ona tylko zrozumieć, ale także stworzyć inteligentne jednostki.

W dzisiejszym świecie Sztuczna Inteligencja jest nieodzownym elementem gier komputerowych i wraz z nimi przez ostatnie dziesięciolecie drastycznie ewoluowała. Można by wręcz powiedzieć, że prawo Moore'a<sup>2</sup> obowiązuje również dla gier video<sup>3</sup>. Czym jednak tak naprawdę jest gra komputerowa? Crawford w swojej książce „*The Art of Computer Game Design*” [5] wyróżnił jej cztery fundamentalne elementy: reprezentacja, interakcja, konflikt i bezpieczeństwo. Gra przedstawia wirtualną rzeczywistość, w której to gracz wzajemnie oddziałuje z wirtualnymi istotami, stawiającymi mu wyzwanie w reprezentowanym przez środowisko gry świecie, bez groźby doznania fizycznego ryzyka. I to właśnie interakcja jest najważniejszym elementem, który wraz z wzrastającą mocą obliczeniową komputerów, bezprecedensowo zdeklasował konwencjonalne gry.

Dwie techniki jakie zastosowaliśmy w zaprojektowanym przez nas modelu Sztucznej Inteligencji w naszej grze komputerowej prezentują, jak można zwiększyć jakość wzajemnego oddziaływanego pomiędzy jednostkami w grze. Pierwszą z technik jest inteligentny agent, dostarczający nowe spojrzenie na architekturę projektowanej gry, prowadząc do bardziej elastycznej interakcji. Agent jest to autonomiczna jednostka, posiadająca sensory, rozumowanie, akcje oraz swoje własne cele. Agenci postrzegani są przez graczy jako inteligentne jednostki, oddziałyujące na siebie językiem

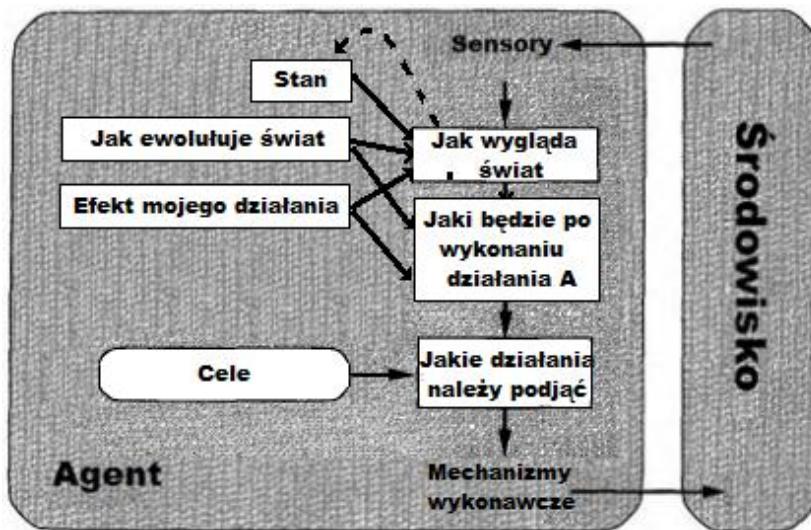
---

<sup>1</sup> Na podstawie słownika łacińskiego „*Lingua Latina Omnibus*” [2].

<sup>2</sup> Prawo empiryczne twierdzące, że optymalna ekonomicznie liczba tranzystorów w układzie scalonym podwaja się w niemalże równych odstępach czasowych [4].

<sup>3</sup> To nasz subiektywny osąd.

komunikacyjnym wysokiego poziomu, a nie poprzez proste interfejsy. Rozproszona natura takiej architektury ułatwia tworzenie złożonych systemów zachowawczych.<sup>4</sup>



Ilustracja 1.1. Inteligentny agent<sup>5</sup>

Schemat przedstawiony na ilustracji 1.1 ukazuje działanie agenta, który w pewnym środowisku monitoruje swoje otoczenie poprzez sensory oraz podejmuje autonomiczne decyzje, wskutek czego wykonuje szereg akcji, aby osiągnąć swój cel.

Drugą z technik wdrożonych do stworzonego przez nas modelu Sztucznej Inteligencji jest logika rozmyta. W kontraste do tradycyjnej logiki, gdzie binarny zbiór posiada dwie wartości logiczne prawda lub fałsz, zmienne logiki rozmytej przyjmują wartości pośrednie pomiędzy stanem 0, a stanem 1, wyrażając poziom przynależności do danego zbioru. Logika rozmyta zapewnia prosty sposób na dojście do określonych wniosków na podstawie niewyraźnych, dwuznacznych, nieprecyzyjnych danych. Celem sterowania logiki rozmytej<sup>6</sup> jest kierowanie złożonymi procesami, przy użyciu strategii kontroli, opartej na wiedzy pochodzącej z ludzkiego doświadczenia. W dziedzinie gier, sterowanie rozmyte jest to system decyzyjny, umożliwiający wyrafinowaną kontrolę zachowania. Jednostki z bardziej subtelnym zachowaniem posiadają zwiększoną percepcję złożoności wirtualnego środowiska.<sup>7</sup>

Tytułowe „AI” tematu pracy „AI w nowoczesnej grafice komputerowej na przykładzie inteligentnego i autonomicznego robota w grze komputerowej”, jest skrótem z angielskiego od „Artificial Intelligence” – Sztuczna Inteligencja.

<sup>4</sup> Opracowane na podstawie S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

<sup>5</sup> Na podstawie ilustracji 2.13 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

<sup>6</sup> Fuzzy Logic Control (ang.) – sterowanie logiki rozmytej .

<sup>7</sup> Opracowane na podstawie J. Sousa, U. Kaymak, „Fuzzy Decision Making in Modeling and Control” [6].

## 1.1 Motywacja

„Kreatywny umysł bawi się obiektyami, które kocha.”<sup>8</sup>

Carl Jung

Pomysł na stworzenie pracy magisterskiej dotyczącej stworzenia autonomicznej jednostki działającej w sztucznie stworzonym środowisku narodził się w dużej mierze dzięki prof. Young Im Cho z Korei Południowej, która to na prośbę władz naszego wydziału poprowadziła przedmiot "Soft Computing", na którym mieliśmy możliwość zapoznania się z tą tematyką, co zaowocowało dalszą chęcią zagłębiania się w ten temat.

Szczegóły naszej pracy zdefiniowaliśmy dopiero w następnym semestrze podczas pobytu na studiach w Portugalii gdzie mogliśmy zagłębić się w tematykę Sztucznej Inteligencji oraz gier komputerowych. Kierowała nami chęć zmierzenia się z problematyką zastosowania logiki rozmytej w symulacjach czasu rzeczywistego, jakimi mogą być gry komputerowe. Podczas zagłębiania się w ten temat, logika rozmyta wydawała nam się znakomitym narzędziem do takich zastosowań przez swoją względną prostotę oraz łatwość mapowania ludzkiej percepcji na percepcję systemu.

Podczas projektowania gry „Halo” pierwszy model agentów bazował na logice rozmytej. Agenci posiadali cztery zmienne rozmyte: strach, złość, obronność, zaskoczenie. Zmienne byłyby modyfikowane poprzez dane zbierane z otoczenia oraz zdarzenia. Jednak projektanci odeszli od tego pomysłu ponieważ okazało się, że było to zbyt trudne do zrozumienia dla graczy. [7]

## 1.2 Cel pracy

Celem niniejszej pracy jest zaprojektowanie oraz implementacja Sztucznej Inteligencji robota w grze komputerowej z wykorzystaniem nowoczesnej grafiki komputerowej. W celu wdrożenia i rozwoju modelu Sztucznej Inteligencji stworzyliśmy grę „Tank Wars”<sup>9</sup>. Początkowa wersja gry została napisana na Uniwersytecie Technicznym w Lizbonie<sup>10</sup> podczas naszych półrocznych studiów zagranicznych w Portugalii<sup>11</sup> pod nadzorem prof. João Madeiras Pereira oraz prof. João da Costa Fernandes. Ideą tematyki stworzonej przez nas gry było przeniesienie starej, konsolowej gry zręcznościowej „Battle City” z 1985 roku<sup>12</sup> do trójwymiarowego świata batalii

<sup>8</sup> Ang. „The creative mind plays with the objects it loves”, Carl Jung, tłumaczenie własne.

<sup>9</sup> Nazwa własna.

<sup>10</sup> Oryginalna nazwa po portugalsku: Universidade Técnica de Lisboa .

<sup>11</sup> Stąd też wywodzi się angielskojęzyczna nazwa gry, przy której postanowiliśmy pozostać.

<sup>12</sup> Na podstawie StrategyWiki [8].

czołgów. Głównymi założeniami fabuły jest czołg – autonomiczna jednostka będąca inteligentnym agentem, której działanie oparte jest na logice rozmytej, poruszająca się po trójwymiarowym świecie, broniąc przed nieprzyjacielskimi jednostkami monumentu orzełka – głównego celu przeciwników. Intencją czołgu – agenta posiadającego wiedzę o środowisku i o sobie, jest zestaw akcji, które musi wykonać, aby osiągnąć swój cel.

Istotną cechą pracy jest jej wizualny aspekt. Przy tworzeniu środowiska gry główny nacisk został nałożony na wygląd modeli obiektów; ich animację; system trójwymiarowych cząsteczek imitujących dym, kurz, czy eksplozję; oraz na fizykę gry obsługującą poruszanie się i kolizje czołgów, tor lotu pocisków i niszczenie murów. Środowiskiem programistycznym, w którym powyżej wymienione aspekty gry zostały zaimplementowane, jest zasugerowany przez portugalskich profesorów Microsoft XNA Game Studio [9].

### 1.3 Zakres pracy

Zakres niniejszej pracy obejmuje:

- omówienie zagadnień Sztucznej Inteligencji oraz przedstawienie istniejących rozwiązań modeli AI w grach o czołgach;
- implementacja środowiska gry, stworzenie modeli i ich animacji, systemu cząsteczek, fizyki gry;
- zaprojektowanie i implementacja inteligentnych agentów jako czołgi poruszające się po mapie gry;
- stworzenie modelu i implementacja systemu decyzyjnego, opartego na logice rozmytej, dla inteligentnych jednostek w grze;
- implementacja algorytmów pomocniczych i testowych;
- wizualizacja wyników;
- krótki film przedstawiający symulację zaimplementowanej Sztucznej Inteligencji.

Rozdział drugi szeroko omawia zagadnienie Sztucznej Inteligencji. Materiały zawarte w tym rozdziale bazują głównie na wiadomościach zawartych w książce S. Russell, P. Norvig, „*Artificial Intelligence: A Modern Approach*”, L. Rutkowski, „*Metody i techniki sztucznej inteligencji*” oraz w J. M. C. Sousa, U. Kaymak, „*Fuzzy Decision Making*

*in Modeling and Control".* Znajdziemy tutaj opis rozwoju Sztucznej Inteligencji na przełomie lat, opracowanie zagadnień dotyczących inteligentnego agenta oraz logiki rozmytej, które tłumaczą podstawy teoretyczne zaprojektowanego przez nas modelu Sztucznej Inteligencji.

Rozdział trzeci omawia podejście do Sztucznej Inteligencji od strony gier komputerowych. Odnosi się do tematyki programowania Sztucznej Inteligencji w grach. Rozdział poświęcony jest przedstawieniu kilku istniejących modeli Sztucznej Inteligencji w grach. Analiza tych modeli pomoże rozszerzyć zakres wiedzy z projektowania Sztucznej Inteligencji.

Rozdział czwarty opisuje świat nowoczesnej grafiki komputerowej od kuchni. Przedstawia jak z prostych wierzchołków i linii powstają realistyczne sceny, nie odbiegające na krok od otaczającego nas świata rzeczywistego.

Rozdział piąty przedstawia koncepcje, według której w grze będzie działać Sztuczna Inteligencja. Informacje zawierają zagadnienia opisujące jej model oraz sposób wdrożenia w naszej grze. W tym rozdziale zaprezentowana jest również idea samej gry.

Rozdział szósty omawia sposób realizacji pracy. Począwszy od środowiska i podstawowych elementów gry, poprzez realizację naszej koncepcji modelu Sztucznej Inteligencji, do przedstawienia wyników symulacji uzyskanego prototypu.

## 2. Sztuczna Inteligencja

„Każdy aspekt nauki lub jakakolwiek inna cecha inteligencji może być tak dokładnie opisana, że urządzenie może to zasymulować.” [10]

John McCarthy

Rozdział wprowadza do tematyki Sztucznej Inteligencji. Przedstawimy w nim próby zdefiniowania pojęcia „*Sztuczna Inteligencja*”, przebieg jej historii i zmian na przełomie lat. Opisane zostaną techniki wykorzystane przez nas do zaprojektowania modelu Sztucznej Inteligencji. W tym, przedstawimy definicje i rodzaje intelligentnych agentów oraz podstawowe założenia logiki rozmytej. Szerzej zapoznamy się z teorią zbiorów rozmytych, opracowaną na podstawie książki Rutkowskiego „*Metody i techniki sztucznej inteligencji*”, na której bazuje rozmyte wnioskowanie zastosowane w naszym modelu Sztucznej Inteligencji.

### 2.1 Czym jest Sztuczna Inteligencja

Nie ma jednoznacznej i ogólnie przyjętej definicji Sztucznej Inteligencji, różnią się one głównie w kilku kwestiach pomiędzy różnymi źródłami. W książce "Artificial Intelligence: A Modern Approach" autorzy podzieliли je na cztery kategorie:

- Systemy, które myślą jak ludzie,
- Systemy, które myślą racjonalnie,
- Systemy, które zachowują się jak ludzie,
- Systemy, które zachowują się racjonalnie.

Jak można zauważyć pierwsze dwie kategorie koncentrują się na procesie myślowym, natomiast dwie kolejne na zachowaniu. Pierwsza i trzecia kategoria skupią się na naśladowaniu ludzi, a pozostałe dwie na dążeniu do modelu idealnej inteligencji, którą można nazwać racjonalnością, a jak wiadomo ludzie nie zawsze są racjonalni, czego przyczyną są najczęściej emocje. [3]

Słynny Test Turinga, zaproponowany przez Alana Turinga w roku 1950, został zaprojektowany by skonstruować satysfakcjonującą definicję inteligencji. Miał sprawdzić czy maszyna z sukcesem zachowuje się jak człowiek tzn. intelligentnie. Pomysł polegał na skonfrontowaniu maszyny z osobnikiem intelligentnym jakim jest istota ludzka. Test

zostałby zaliczony pozytywnie, jeżeli po zadaniu przez operatora kilku pytań maszynie, nie mógłby on stwierdzić czy uzyskane odpowiedzi pochodzą od maszyny, czy od człowieka. Istnieje również tak zwany kompletny Test Turinga, który do testu wprowadza wymagania związane z fizyczną interakcją z operatorem. [1]

Dział nauki „*Kognitywistyka*” zajmuje się „*myśleniem jak ludzie*”. Bazuje on na teoriach naukowych opisujących sposób działania umysłu człowieka. U jego podstaw leży założenie, że powstanie wystarczająco precyzyjnej teorii definiującej funkcjonowanie umysłu, pozwoli przełożyć ją na komputerowe programy. Jednakże obecnie posiadana teoria tego działu daleko jest od wyjaśnienia sposobu działania ludzkiej inteligencji. [11]

Inną możliwością jest poszukiwanie "praw myślenia", grecki filozof Arystoteles jako jeden z pierwszych podjął próbę skodyfikowania "właściwego myślenia", jego sylogizm udostępnił wzory konstrukcji, których wnioski były zawsze właściwe po podaniu poprawnych przesłanek np. Sokrates jest mężczyzną, wszyscy mężczyźni są śmiertelni, dlatego Sokrates jest śmiertelny. Takie prawa myślenia miały kierować umysłem, studiowanie ich zapoczątkowało dziedzinę zwaną logiką. Jednak istnieją dwie główne przeszkode przy takim podejściu, pierwsza to trudność zapisania nieformalnej wiedzy w formie formalnych wyrażeń wymaganych przez notację logiki, w szczególności gdy wiedza nie jest w stu procentach pewna. Drugą przeszkode jest praktyka, problemy złożone z kilkudziesięciu faktów mogą przeciążyć komputer jeżeli nie podamy mu pewnych wskazówek dotyczących kolejności podejmowanych kroków wnioskowania. [3]

Działanie w sposób racjonalny jest bardziej ogólne niż myślenie racjonalne, ponieważ bycie racjonalnym nie musi polegać głównie na logicznych rozważaniach, ale może również na refleksie. Ponadto istnieją sytuacje, w których nie da się dowieść poprawności konkretnych działań, mimo to jakaś czynność musi zostać podjęta. Zaletą rozpatrywania sztucznej inteligencji w kontekście racjonalnych zachowań, jest większa podatność na rozwój naukowy, niż na koncepcje bazujące na ludzkim myśleniu oraz działaniu. Racjonalność jest jasno zdefiniowana, natomiast ludzki umysł oraz zachowanie, wynikają z długiego oraz skomplikowanego procesu ewolucyjnego, przystosowującego człowieka do konkretnego środowiska. Ludzka inteligencja jest wciąż daleka od perfekcji. Głównym podmiotem takiego podejścia jest agent racjonalny. [3]

Swój wkład w rozwój Sztucznej Inteligencji nieustannie ma wiele dziedzin nauk i technik takich jak: Filozofia, Matematyka, Ekonomia, Neurobiologia, Psychologia, Informatyka, Teoria Sterowania oraz Lingwistyka. Na jej komponenty składają się samodzielne dyscypliny mające podobne cele, takie jak teoria sterowania, teoria podejmowania decyzji, czy badania operacyjne. Jednak Sztuczna Inteligencja została uznana za nową dyscyplinę ponieważ jak twierdzą S. Russell oraz P. Norvig w swojej książce [3, s. 18] „*Sztuczna Inteligencja od początku uwzględniała zamiar skopiowania zdolności ludzkich takich jak kreatywność, samodoskonalenie oraz komunikację za pomocą języka.*” Żadna z poszczególnych dyscyplin tego nie obejmowała.

## 2.2 Rozwój Sztucznej Inteligencji na przełomie lat

Załążki rozwoju Sztucznej Inteligencji miały miejsce już w czasach starożytności, jednak to data uznana za jej narodziny jak i gwałtowny jej rozwój przypada na drugą połowę XX wieku. Czas ten dzieli się odpowiednio na następujące okresy:

### ▪ Początki Sztucznej Inteligencji

- **1943** W. McCulloch i W. Pitts proponują model sztucznych sieci neuronów. Jest to pierwsza uznawana praca o tematyce Sztucznej Inteligencji. Neuron posiada wiele wejść i jedno wyjście, może znajdować się w stanie wyłączonej lub włączonej z możliwością zmiany w wyniku oddziaływania odpowiedniej liczby sąsiednich neuronów;
- **1950** A. Turing opisuje wizje Sztucznej Inteligencji w artykule „*Computing Machinery and Intelligence*”;
- **1951** M. Minsky i D. Edmonds budują pierwszą sztuczną sieć neuronową symulującą szczura szukającego wyjścia z labiryntu;
- **1965** J. McCarthy nazywa nową dyscyplinę „*Sztuczna Inteligencja*” podczas konferencji w Dartmouth College, USA.

### ▪ Wczesny entuzjazm i wielkie oczekiwania

- **1957** H. Simon stwierdza: „stworzymy w ciągu dziesięciu lat maszyny wnioskujące i myślące”
- **1950 – 60** Okres powstawania pierwszych programów Sztucznej Inteligencji: A. Samuel tworzy serie programów do gry w warcaby, które nauczyły się grać na wysokim poziomie amatorskim, obalając pogląd, że komputery mogą robić tylko to, co się im każe. H. Gelernter tworzy program „*Geometry Theorem Prover*” zdolny do rozwiązywania zadań, których wielu studentów matematyki uznało by za dość trudne;
- **1965** J. Robinson tworzy algorytm logicznego wnioskowania, który dowódź twierdzenie, z którym matematycy nie mogli sobie poradzić;
- **1950 – 67** Znaczący postęp pracy nad sieciami neuronowymi.

### ▪ Pesymizm i realizm

- **1966 – 73** Sztuczna Inteligencja zmaga się z barierą złożoności obliczeniowej;
- **1966** Pesymistyczne nastawienie w problematyce automatycznego tłumaczenia tekstów. Wczesne programy Sztucznej Inteligencji wymagały bardzo niewiele lub nie wymagały wcale wiedzy związanej z danym tematem, natomiast tłumaczenie tekstów wymaga ogólnej wiedzy na odnośnie składni języków by uniknąć dwuznaczności oraz utworzyć poprawne składniowo zdanie. Jednym ze słynnych przykładów ilustrujących ten problem jest zdanie „*The spirit is willing but the flesh is weak*”, które

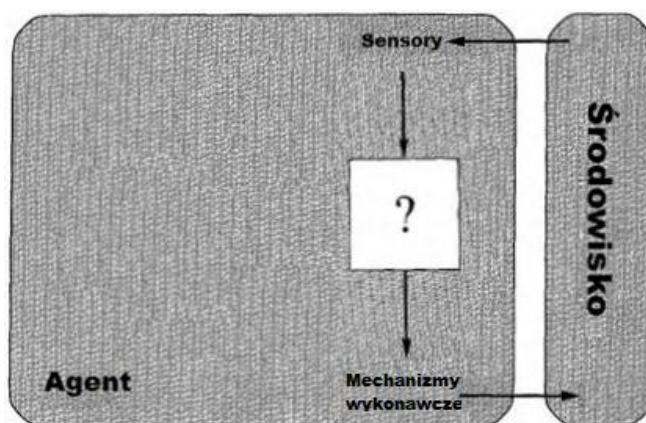
z języka angielskiego zostało przetłumaczone na język rosyjski oraz ponownie na angielski. W rezultacie otrzymano zdanie „*The vodka is good but the meat is rotten*”.

- **1969** Zaprzestanie badań nad sieciami neuronowymi.
- **Wyjście z zastoju i rozwój**
  - **1969 – 79** Pierwsze bazy wiedzy: „*DENDRAL*” o chemii, „*MYCIN*” o medycynie, „*SHRDLU*” o lingwistyce;
  - **1980** Wkroczenie Sztucznej Inteligencji do przemysłu. Początek zastosowania systemów ekspertowych w firmach;
  - **1987** Formalizacja metod i teorii Sztucznej Inteligencji jako nauki;
  - **1995** Pierwsze inteligentne agenty; „*SORA*” pierwszy agent naśladujący myślenie człowieka. Powstanie pierwszych „Botów” internetowych.

Aktualnie Sztuczna Inteligencja towarzyszy nam w codziennym życiu. Znajduje ona swoje zastosowanie w biznesie, ekonomii, prognozowaniu, diagnostyce medycznej, planowaniu autonomicznym, logistycznym, robotyce, dowodzeniu twierdzeń, rozumieniu i tłumaczeniu języków, oraz różnych systemach ekspertowych. [3]

### 2.3 Inteligentny agent

Agentem może być wszystko to, co jesteśmy w stanie postrzegać jako obiekt potrafiący zbierać informacje ze środowiska oraz na ich podstawie podejmować działanie. Interakcja ze środowiskiem odbywa się poprzez system czujników, które zbierają potrzebne informacje, oraz system mechanizmów wykonawczych, za pomocą których możliwe jest podjęcie charakterystycznych dla danego agenta działań.

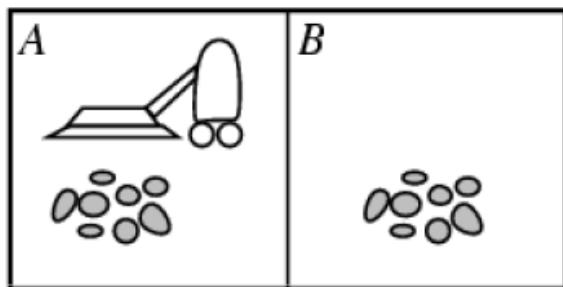


Ilustracja 2.1. Interakcja agenta ze środowiskiem.<sup>13</sup>

<sup>13</sup> Na podstawie ilustracji 2.1 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

Do połączenia tych systemów potrzebna jest funkcja mapująca, która charakteryzuje zachowanie agenta (Ilustracja 2.1). W najprostszym przypadku, funkcją mapującą może być zwykła tabela, łącząca każdą możliwą sekwencję zebranych informacji z odpowiadającym jej zestawem akcji. Decyzja agenta może być podejmowana również na podstawie historii obserwacji środowiska, jednakże w tym przypadku problemem mogą okazać się jej nieskończanie duże rozmiary. [3]

Do zilustrowania prostego agenta posłuży przykład odkurzacza, który działa w środowisku złożonym z dwóch pomieszczeń. Każde pomieszczenie może być czyste lub nie. Zadaniem agenta jest utrzymanie czystości w obu pomieszczeniach. Informacje jakie może pozyskać ze środowiska to własne położenie oraz status czystości miejsca, w którym się znajduje. Działania, które może podjąć to bezczynność, odkurzanie oraz przemieszczenie się w lewo lub prawo. [3]



Ilustracja 2.2. Przykład agenta odkurzacza.<sup>14</sup>

Tabela 1 przedstawia działania agenta zależne od sekwencji obserwacji. Ze względu na możliwie nieskończoną liczbę różnych sekwencji, które może zaobserwować agent, rozmiar tablicy działań może być równie nieskończony.

Tabela 1. Działania agenta odkurzacza<sup>15</sup>

Sekwencja obserwacji	Działanie
[A, Czysto]	Przemieszcz się w prawo
[A, Brudno]	Odkurzaj
[B, Czysto]	Przemieszcz się w lewo
[B, Brudno]	Odkurzaj
[A, Czysto], [A, Czysto]	Przemieszcz się w prawo
[A, Czysto], [A, Brudno]	Odkurzaj
...	...
[A, Czysto], [A, Czysto], [A, Czysto]	Przemieszcz się w prawo
[A, Czysto], [A, Czysto], [A, Brudno]	Odkurzaj
...	...

<sup>14</sup> Na podstawie ilustracji 2.2 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

<sup>15</sup> Na podstawie ilustracji 2.3 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

## ■ Miara skuteczności

Agentem racjonalnym może być nazwany ten, który podejmuje właściwe działanie w danej sytuacji tzn. takie, które sprawia, że agent odnosi największy sukces. Dlatego potrzebna jest możliwość pomiaru odniesionego sukcesu. Miarą skuteczności agenta są kryteria, którym podlega jego zachowanie w celu oceny jego działania. Gdy agent jest wpuszczony do danego środowiska, generuje sekwencje zachowań uzależnionych od obserwacji. Ta sekwencja sprawia, że środowisko przechodzi przez sekwencje stanów. Jeżeli z kolei ta sekwencja będzie satysfakcjonującą, to oznacza, że agent działa dobrze. Oczywiście nie ma jednej uniwersalnej miary odpowiedniej dla wszystkich agentów. Każdy typ należy rozpatrywać indywidualnie, tak by wybrane kryteria pozwalały ocenić go obiektywnie. [3]

Powracając do przykładu odkurzacza, miarą skuteczności może być ilość posprzątanego brudu w czasie jednej ósmiodziesięcioletniej zmiany. Jednak agent racjonalny może zmaksymalizować swoją skuteczność poprzez sprzątanie, a następnie upuszczenie na podłogę tego co zebrał i ponowne sprzątanie. Odpowiedniejszą miarą byłoby wynagradzanie agenta za czystą podłogę. Dla przykładu jeden punkt mógłby być przyznany za jeden czysty pokój w jednym kroku czasowym. W ogólności lepiej projektować miarę jakości według tego czego oczekujemy w danym środowisku, niż tego jak agent powinien się zachowywać. [3]

## ■ Racjonalność

To co jest racjonalne w danej chwili zależy od czterech czynników:

- Miary skuteczności, która definiuje kryteria sukcesu agenta,
- Jego poprzedniej wiedzy na temat środowiska,
- Działań, które może podjąć,
- Obserwacji dokonanych do danej chwili.

Na podstawie tych czynników powstała definicja racjonalnego agenta: "*Dla każdej możliwej sekwencji obserwacji agent racjonalny powinien podjąć taką akcję, która maksymalizuje jego miarę skuteczności na podstawie uzyskanych obserwacji oraz wiedzy, którą posiadał.*" [3]

To czy ten agent jest racjonalny zależy od paru czynników. Najpierw należy zdefiniować jaka będzie jego miara skuteczności, jaką wiedzę o środowisku posiada, jakie ma do dyspozycji czujniki oraz jakie działania może podjąć. Wracając do przykładu agenta odkurzacza, aby stwierdzić, że jest on racjonalny, należy przyjąć następujące założenia:

- Miara skuteczności polega na wynagrodzeniu agenta jednym punktem za każde pomieszczenie utrzymane w czystości w jednym kroku czasowym przez okres 1000 kroków;
- Rozkład pomieszczeń w środowisku jest wiadomy, lecz dystrybucja brudu w pomieszczeniach oraz początkowa lokalizacja agenta już nie. Odkurzanie czyści pomieszczenie, w którym znajduje się agent, natomiast czyste pomieszczenie pozostaje czyste. Agent może przemieszczać się w lewo lub w prawo, za wyjątkiem sytuacji wyjścia poza środowisko, wtedy pozostaje w miejscu;
- Jedyne możliwe działania to: przemieść się lewo lub w prawo, odkurzaj, stój bezczynnie;
- Agent zna swoje położenie oraz czy tam gdzie się znajduje jest brudno.

Takiego agenta można uznać za racjonalnego. Należy również rozróżnić racjonalność od wszechwiedzy. Wszechwiedzący agent zna rezultat swojego działania i może według niego podejmować decyzje. Jednak bycie wszechwiedzącym jest w rzeczywistości niemożliwe do osiągnięcia. Definicja racjonalnego agenta nie wymaga wszechwiedzy, ponieważ racjonalna decyzja zależy głównie od sekwencji obserwacji zdobytych do danej chwili. Agent powinien również zbierać informacje z otoczenia oraz uczyć się na podstawie własnych działań. Są również przypadki gdzie środowisko jest z góry dobrze znane, w takich okolicznościach, agent nie musi zbierać informacji lub uczyć się, po prostu zachowuje się poprawnie. [3]

## ■ Środowisko

Racjonalnego agenta można nazwać "*rozwiązaniem*", a środowisko zadania, dla którego został stworzony jest "*problemem*", który należy rozwiązać. Na środowisko zadania składają się cztery czynniki, są to: miara skuteczności, środowisko, czujniki oraz mechanizm wykonawczy. Przy projektowaniu agenta, pierwszym krokiem jaki należy podjąć jest sprecyzowanie tych elementów najdokładniej jak się da.

Zakres różnych typów środowisk jakie mogą się pojawić w problemach Sztucznej Inteligencji jest olbrzymi, jednak można je skategoryzować według następujących właściwości:

- W pełni obserwowlane (lub częściowo obserwowlane) - agent w każdej chwili może pozyskać pełne informacje na temat całego środowiska za pomocą czujników;
- Deterministyczne (lub stochastyczne) - kolejny stan środowiska jest uzależniony tylko od bieżącego stanu oraz działania agenta;
- Chwilowe (lub sekwencyjne) - działanie agenta jest podzielone na kroki, każdy krok polega na dokonaniu obserwacji i wykonaniu działań. Podjęte działanie w każdym kroku zależy tylko od tego kroku;

- Statyczne (lub dynamiczne) - środowisko jest niezmienne w trakcie działania agenta;
- Dyskretne (lub ciągłe) - składa się z przeliczalnej liczby obserwacji i działań;
- Pojedynczy agent (lub wielu agentów) - w środowisku działa tylko jeden agent.

Tabela 2 prezentuje na przykładach wymieniony powyżej własności.

Tabela 2. Przykłady i charakterystyka środowisk zadania.<sup>16</sup>

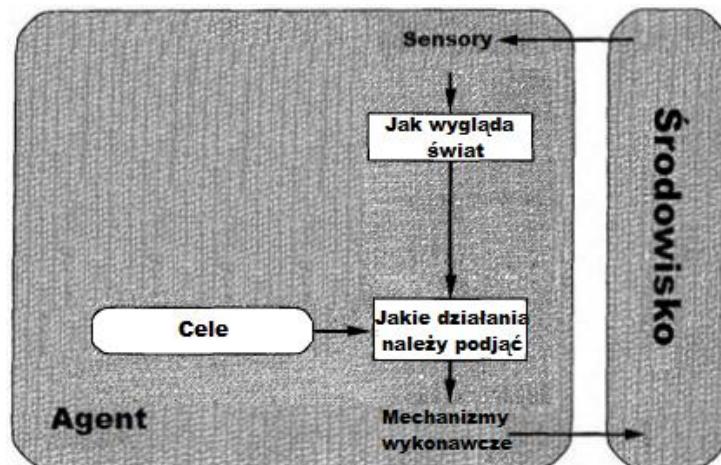
Środowisko Zadania	Obserwowałe	Deterministyczne	Chwilowe	Statyczne	Dyskretne	Agent
<b>Szachy z zegarem</b>	W pełni	Strategiczne	Sekwencyjne	Częściowo	Dyskretne	Wielu
<b>Krzyżówka</b>	W pełni	Deterministyczne	Sekwencyjne	Statyczne	Dyskretne	Jeden
<b>Poker</b>	Częściowo	Stochastyczne	Sekwencyjne	Statyczne	Dyskretne	Wielu
<b>Diagnoza medyczna</b>	Częściowo	Stochastyczne	Sekwencyjne	Dynamiczne	Ciągłe	Jeden
<b>Analiza obrazu</b>	W pełni	Deterministyczne	Chwilowe	Częściowo	Ciągłe	Jeden
<b>Robot zbierający</b>	Częściowo	Stochastyczne	Chwilowe	Dynamiczne	Ciągłe	Jeden
<b>Interaktywny nauczyciel angielskiego</b>	Częściowo	Stochastyczne	Sekwencyjne	Dynamiczne	Dyskretne	Wielu

Najtrudniejszym przypadkiem jest środowisko częściowo obserwowałe, stochastyczne, sekwencyjne, dynamiczne, ciągłe z wieloma agentami. Okazuje się również, że większość realnych sytuacji jest tak skomplikowana, że określenie, czy są deterministyczne jest punktem spornym. W praktyce muszą być traktowane jako stochastyczne. [3]

### ▪ Agent refleksyjny

Działania agenta refleksyjnego są podejmowane wyłącznie na podstawie aktualnych obserwacji, ignorując ich historię oraz wbudowane reguły. Takim agentem jest robot odkurzacz z poprzedniego przykładu. Refleksywne działanie redukuje liczbę możliwych sytuacji. Jest ono realizowane za pomocą zasady „warunek - działanie” określonej przez cel agenta. Dla przykładu odkurzacza warunkiem jest „Brudno”, a działaniem „Odkurzaj”. Ilustracja 2.3 przedstawia strukturę takiego agenta. Agent refleksowy ma tą właściwość, że jest prosty przez co ma ograniczone możliwości bycia inteligentnym. Będzie dobrze działał tylko wtedy gdy możliwe jest podjęcie poprawnej decyzji tylko na podstawie bieżących obserwacji tzn. wtedy gdy środowisko jest w pełni obserwowałe. Nawet niewielki stopień zmniejszający obserwowałość środowiska może sprawić spore problemy. W przypadku odkurzacza, jeżeli posiadałby on tylko czujnik rozpoznający, czy pomieszczenie jest brudne, a nie mógłby on określić swojego

<sup>16</sup> Na podstawie ilustracji 2.6 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].



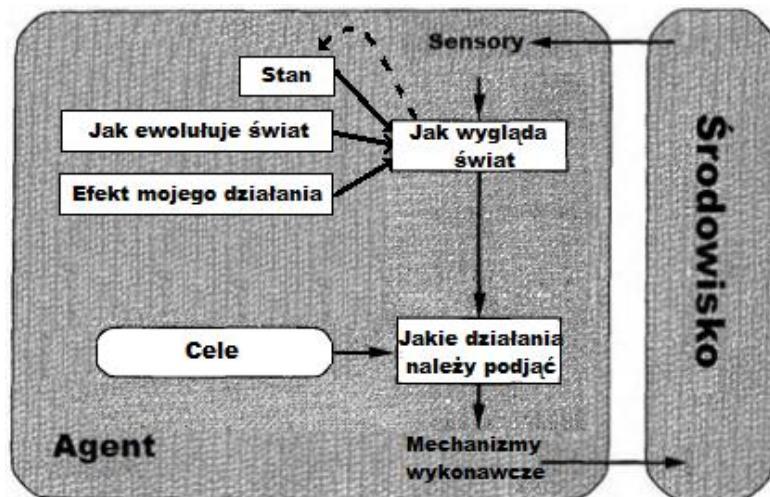
Ilustracja 2.3. Struktura agenta refleksyjnego.<sup>17</sup>

położenia, czyli nie posiadałby wszystkich informacji o środowisku, potrzebnych mu do jego prawidłowego działania. Jedyne obserwacje jakich mógł by dokonać to „Czysto” lub „Brudno”. Jeżeli w pomieszczeniu będzie czysto, to agent ma możliwość przemieszczenia się w lewo lub prawo. Jednak jeżeli będzie w pomieszczeniu A, to przemieszczenie w lewo zawiedzie, a gdy będzie w pomieszczeniu B, to przemieszczenie w prawo zawiedzie. Przypomnieć należy, tego typu agent nie pamięta w jakim wcześniej był pomieszczeniu. Nieskończone pętle są często nieuniknione dla agenta refleksyjnego działającego w częściowo obserwowałyym świecie. Wyjściem z tej sytuacji mogłoby być losowe wybieranie kierunku. [3]

#### ■ Agent refleksyjny z modelem świata

Model świata agenta refleksyjnego służy do symulacji środowiska nie w pełni obserwowałoego. Dzięki niemu agent jest w stanie wyznaczyć przybliżony stan środowiska, w którym się znajduje, poprzez uzupełnianie brakujących o nim informacji. Dzięki takiemu modelowi możliwe jest przewidywanie w środowisku zmian wywołanych działaniem agenta. By uporać się z częściową obserwowałością środowiska, agent musi śledzić stan części środowiska, której aktualnie nie widzi. Powinien on posiadać wewnętrzny stan, który zależy od historii obserwacji, dzięki czemu poznaje wiedzę na temat pewnych, nieobserwowanych aspektów obecnego stanu. By tego dokonać, agent musi oprócz zapamiętanego stanu posiadać dwa rodzaje wiedzy. Pierwszy z nich to wiedza na temat tego jak działa świat, np. po jakim czasie pomieszczenie znowu będzie brudne. Drugi rodzaj, to jakie będą efekty działań podjętych przez agenta. Jeżeli przemieści się w lewo a wcześniej był w pomieszczeniu B, to znajdzie się w pomieszczeniu A. Ilustracja 2.4 przedstawia strukturę agenta refleksyjnego z modelem na świat. [3]

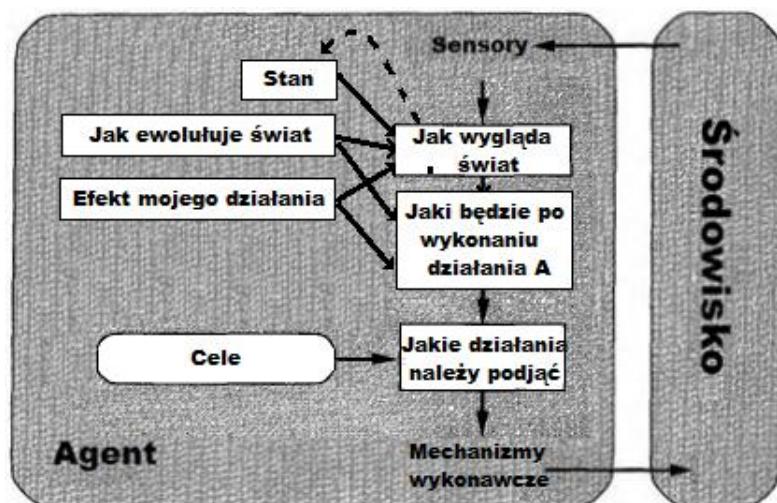
<sup>17</sup> Na podstawie ilustracji 2.9 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].



Ilustracja 2.4. Struktura agenta refleksyjnego z modelem na świat.<sup>18</sup>

#### ▪ Agent celowy

Agent ma z góry wyznaczony cel, który definiuje pożądane stany. Może lepiej wykorzystać posiadane informacje o świecie niż agent refleksowy z modelem świata. Dysponowaną wiedzą na temat efektów swoich działań, może skonfrontować ze swoim celem i sprawdzić, które действие najbardziej przybliży go do tego celu. W procesie decyzyjnym już nie ma reguł typu „warunek - działanie”, tak jak to było w przypadku agenta refleksyjnego, przez które był on przywiązany do prostego mapowania obserwacji na действие.



Ilustracja 2.5. Struktura agenta celowego.<sup>19</sup>

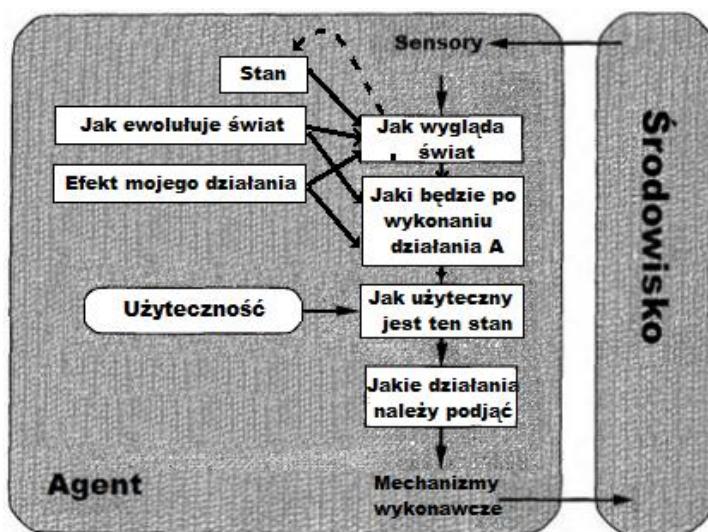
<sup>18</sup> Na podstawie ilustracji 2.11 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

<sup>19</sup> Na podstawie ilustracji 2.13 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

Agent celowy, którego strukturę prezentuje Ilustracja 2.5, jest mniej wydajny, ale za to bardziej elastyczny. Może modyfikować swoją wiedzę na temat środowiska, co jest odzwierciedlane w jego informacji na temat rezultatów odpowiednich działań. Dodatkowo cel takiego agenta może zostać łatwo zmodyfikowany. Reguły agenta refleksyjnego pozwalają mu podążać w jednym zakodowanym celu. Tego typu agent jest najczęściej wykorzystywany w zadaniach poszukiwania oraz planowania. [3]

### ▪ Agent z funkcją użyteczności

Cel jako taki nie jest wystarczający by w większości środowisk generować wysokiej jakości zachowania. Czasem do celu prowadzi więcej niż jedna sekwencja działań, która jest lepsza lub gorsza. Agent celowy potrafi rozróżnić tylko, czy dane действие go satysfakcjonuje, czy nie. Bardziej ogólna miara skuteczności powinna pozwalać na porównanie poszczególnych stanów środowiska oraz rozróżnienie wszystkich tych, które satysfakcjonują agenta by wybrać najkorzystniejszy. Funkcja użyteczności mapuje stan lub ich sekwencję na liczbę rzeczywistą, która opisuje stopień zadowolenia agenta. Funkcja ta pozwala również agentowi na poradzenie sobie z wyborem pomiędzy sprzecznymi celami oraz pomiędzy wieloma możliwymi celami. Ilustracja 2.6 przedstawia wewnętrzny schemat działania takiego agenta. [3]



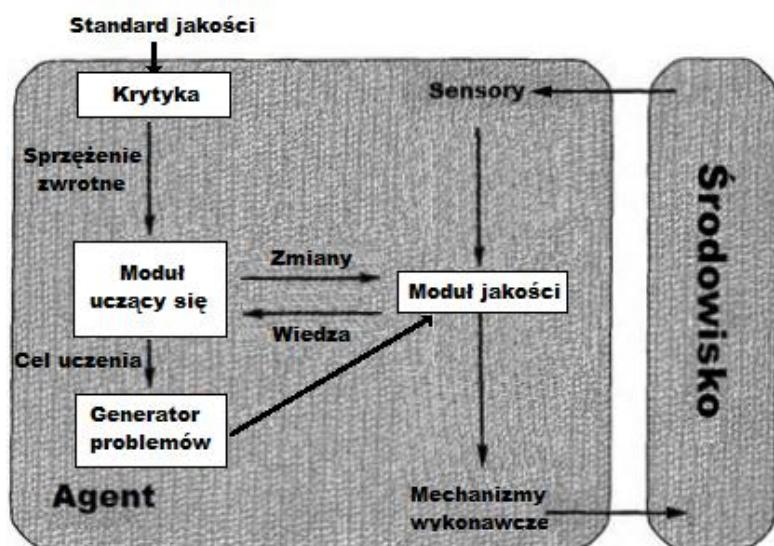
Ilustracja 2.6. Struktura agenta z funkcją użyteczności.<sup>20</sup>

### ▪ Agenci uczący się

Agent uczący się potrafi modyfikować swoją wiedzę o środowisku i pożądanych sposobach działania. Uczyć może się agent każdego typu, dzięki czemu ma możliwość

<sup>20</sup> Na podstawie ilustracji 2.14 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

działania w początkowo nieznanym środowisku lub możliwość poprawienia skuteczności, jeżeli początkowa wiedza nie pozwala na jej maksymalizację. Najbardziej znaczącymi elementami struktury agenta są moduł uczący, odpowiadający za ulepszanie agenta oraz moduł jakości, który odpowiedzialny jest za wybór odpowiednich działań. Moduł jakości jest w pewnym sensie tym, co cały agent z poprzednich paragrafów - pobiera dokonane obserwacje, po czym decyduje jakie działanie należy podjąć. Moduł uczenia używa sprzężenia zwrotnego z krytyki o tym, jak agent radzi sobie w środowisku, oraz determinuje w jaki sposób moduł jakości powinien być zmodyfikowany by podejmować lepsze decyzje w przyszłości. Ostatnim komponentem jest generator problemów. Jego zadanie polega na sugerowaniu działań pozwalających na zdobycie nowych doświadczeń. Moduł jakości działa w sposób jaki został do tego stworzony, jednak czasem agent może chcieć powiedzieć i podjąć nie do końca optymalne akcje, przez co może odkryć nowe możliwości, które w ogólności pozwolą wykonać lepsze działania. Ilustracja 2.7 przedstawia schemat działania agenta uczącego się. [3]



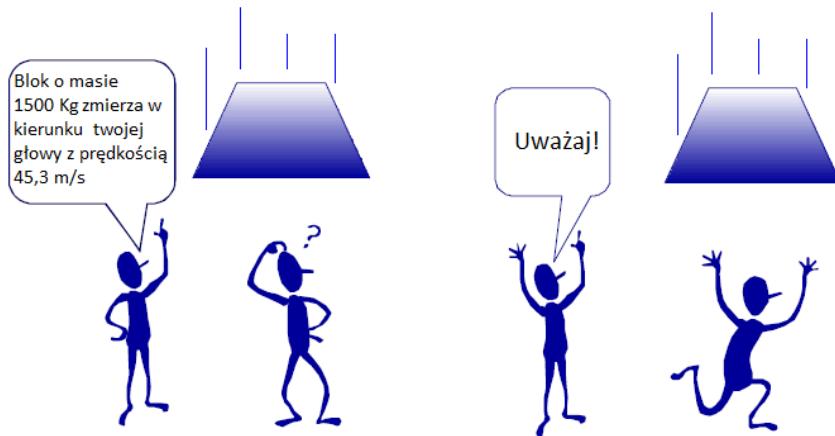
Ilustracja 2.7. Struktura agenta uczącego się.<sup>21</sup>

## 2.4 Logika rozmyta

Logika Rozmyta jest równoznaczna z teorią zbiorów rozmytych, gdzie klasyfikacja obiektów nie ma sztywnych granic, lecz jest kwestią opisu poprzez stopień przynależności. Podstawową koncepcją na której bazuje Logika Rozmyta jest zmienna językowa, czyli zmienna, której wartościami mogą być słowa a nie liczby. Dlatego znaczna jej część może być postrzegana jako metodologia obliczeń z użyciem słów zamiast liczb. Chociaż słowa są mniej precyzyjne niż liczby, ich użycie jest bliższe ludzkiej

<sup>21</sup> Na podstawie ilustracji 2.15 z S. Russell, P. Norvig „Artificial Intelligence: A Modern Approach” [3].

intuicji. Ponadto obliczenia przy użyciu słów wykorzystują tolerancję dla precyzji i dlatego też zmniejszają koszt rozwiązania. [12]



Ilustracja 2.8. Precyza przeciwko Trafność – znacznie i wymowa w logice rozmytej.<sup>22</sup>

Logika Rozmyta dotyczy względnego znaczenia precyzji. Jak ważne jest być dokładnie poprawnym, gdy przybliżona odpowiedź wystarczy. Jest dobrym balansem pomiędzy precyzją, a znaczeniem - czymś, czym ludzie dobrze posługują się od bardzo dawna. Jest również wygodnym narzędziem do odwzorowywania przestrzeni wejściowej do przestrzeni wyjściowej. [12]

Logiki rozmytej warto używać gdyż:

- jest konceptualnie łatwa do zrozumienia,
- elastyczna, łatwo o zmiany, bez konieczności tworzenia jej na nowo,
- posiada wysoki stopień tolerancji dla nieprecyzyjnych danych,
- radzi sobie z niejednoznacznością i sprzecznościami,
- posiada zdolność uogólniania wiedzy. [12]

Podstawą teoretyczną logiki rozmytej są zbiory rozmyte, opisane w następnym dziale.

## 2.5 Podstawowe definicje i właściwości Zbiorów Rozmytych<sup>23</sup>

W tym dziale znajdują się definicję oraz podstawowe pojęcia teorii zbiorów rozmytych wraz z krótkimi przykładami ich zastosowania. Teoria wraz ze wzorami i wykresami w całości pochodzi z książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

<sup>22</sup>Opracowane na podstawie ilustracji z wykładu J.M.C. Sousa „Fuzzy sets” [12].

<sup>23</sup>Dział opracowany na podstawie rozdziału 4 książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

*inteligencji*". Dział ten przedstawia teorię, na które bazuje zaprojektowany przez nas model rozmytego wnioskowania.

Zbiorem rozmytym A, w pewnym niepustym obszarze rozważań X, co zapisujemy jako  $A \subseteq X$ , nazywamy zbiór par:

$$A = \{(x, \mu_A(x)); x \in X\} \quad (1)$$

gdzie:

$\mu_A: X \rightarrow [0, 1]$  jest funkcją przypisującą każdemu elementowi  $x$  stopień przynależności do zbioru A;

$\mu_A(x) = 1$  oznacza pełną przynależność elementu x do zbioru A, tzn.  $x \in A$ ;

$\mu_A(x) = 0$  oznacza brak przynależności elementu x do zbioru A, tzn.  $x \notin A$ ;

$0 < \mu_A(x) < 1$  oznacza częściową przynależność elementu x do zbioru A.

Innym sposobem na zapisu zbioru rozmytego dla przestrzeni  $X$ , ze skończoną liczbą elementów  $X = [x_1, \dots, x_n]$  jest:

$$A = \sum_{i=1}^n \frac{\mu_A(x_i)}{x_i} \quad (2)$$

Przy czym, elementami  $x_i$  oprócz liczb mogą być osoby, przedmioty lub inne pojęcia. Powyższy zapis posiada charakter symboliczny, a kreska ułamkowa nie oznacza operacji dzielenia, lecz przyporządkowanie poszczególnym elementom  $x_i$  stopni przynależności  $\mu_A(x_i)$ . Zapis  $\frac{\mu_A(x_i)}{x_i}$  oznacza parę  $(x, \mu_A(x))$ . Również znak sumy nie oznacza operacji dodawania, lecz sumę mnogościową elementów. Dla przestrzeni rozważań X o nieskończonej liczbie elementów, zbiór A może być symbolicznie zapisany jako:

$$A = \int_x \frac{\mu_A(x)}{x} \quad (3)$$

Dla przykładu<sup>24</sup> założmy, że  $X = N$  jest zbiorem liczb naturalnych. Definicja zbioru liczb naturalnych „bliskich liczby 7” jest następująca:

$$A = \frac{0,2}{4} + \frac{0,5}{5} + \frac{0,8}{6} + \frac{1}{7} + \frac{0,8}{8} + \frac{0,5}{9} + \frac{0,2}{10}. \quad (4)$$

Pamiętamy, że wartość w liczniku jest stopniem przynależności, wartość w mianowniku elementem zbioru liczb naturalnych, a operacje kreski ułamkowej i sumy są zapisem symbolicznym i nie mają zastosowania. Jeżeli przestrzeń X będzie zbiór liczb rzeczywistych, to dla tego samego przykładu, zbiór liczb rzeczywistych „bliskich liczby 7” może posiadać funkcję przynależności w postaci:

$$\mu_A(x) = \frac{1}{1 + (x - 7)^2}. \quad (5)$$

---

<sup>24</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Sam zapis zbioru rozmytego liczb rzeczywistych „*bliskich liczby 7*” przyjmuje postać:

$$A = \int_x \frac{[1 + (x - 7)^2]^{-1}}{x}. \quad (6)$$

Zbiory rozmyte w obu przypadkach można zapisać na wiele sposobów. Na przykład funkcję przynależności może posiadać następującą postać:

$$\mu_A(x) = \begin{cases} 1 - \sqrt{\frac{|x - 7|}{3}}, & \text{jeżeli } 4 \leq x \leq 10, \\ 0, & \text{w przeciwnym razie.} \end{cases} \quad (7)$$

#### ■ Podstawowe funkcje przynależności

W wielu przypadkach zastosowanie znajdują standardowe funkcje przynależności. Poniżej zaprezentowane są ich wzory oraz graficzna reprezentacja.

- Singleton jest funkcją przynależności charakteryzującą jedno elementowy zbiór rozmyty, element ten w pełni należy do zbioru rozmytego, w pozostałych punktach funkcja przyjmuje wartość 0. Elementem w pełni należącym do zbioru A jest punkt  $x_0$ . Tego typu funkcja wykorzystywana jest głównie do realizowania operacji rozmywania, mającej swoje zastosowanie w rozmytych systemach wnioskujących.

$$\mu_A(x) = \begin{cases} 1, & \text{jeżeli } x = x_0, \\ 0, & \text{jeżeli } x \neq x_0. \end{cases} \quad (8)$$

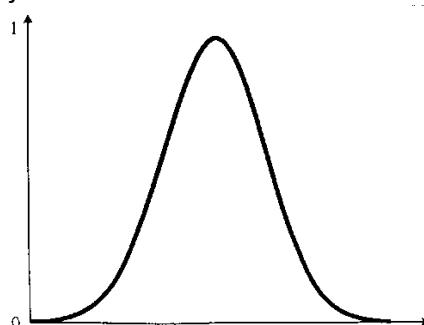
Gaussowska funkcja przynależności jest najczęściej spotykaną. Opisana jest następującym wzorem:

$$\mu_A(x) = \exp\left(-\left(\frac{x - x_0}{\sigma}\right)^2\right), \quad (9)$$

gdzie:

$x_0$  - środek funkcji,

$\sigma$  - określa szerokość krzywej.



Ilustracja 2.9. Gaussowska funkcja przynależności.<sup>25</sup>

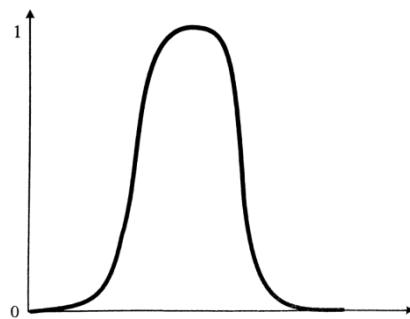
<sup>25</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

- Funkcja przynależności typu dzwonowego jest postaci:

$$\mu_A(x; a, b, c) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}}, \quad (10)$$

gdzie:

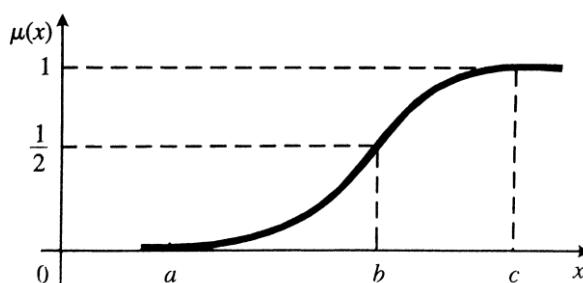
- a - określa szerokość funkcji,
- b - określa jej nachylenie,
- c - określa jej środek.



Ilustracja 2.10. Funkcja przynależności typu dzwonowego.<sup>26</sup>

- Funkcja przynależności klasy s scharakteryzowana jest jako:

$$s(x; a, b, c) = \begin{cases} 0, & \text{dla } x \leq a, \\ 2 \left( \frac{x-a}{c-a} \right)^2, & \text{dla } a < x \leq b, \\ 1 - 2 \left( \frac{x-c}{c-a} \right)^2, & \text{dla } b < x \leq c, \\ 1, & \text{dla } x > c. \end{cases} \quad (11)$$



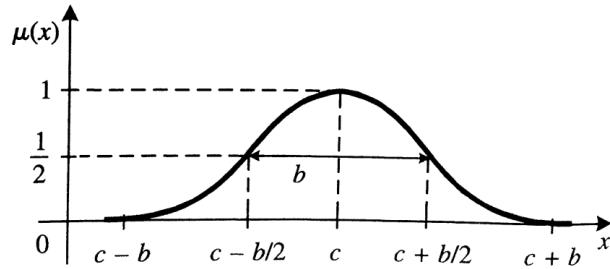
Ilustracja 2.11. Funkcja przynależności klasy s.<sup>24</sup>

- Funkcja przynależności klasy  $\pi$

$$\pi(x; b, c) = \begin{cases} s \left( x; c - b, c - \frac{b}{2}, c \right), & \text{dla } x \leq c, \\ 1 - s \left( x; c, c + \frac{b}{2}, c + b \right), & \text{dla } x > c. \end{cases} \quad (12)$$

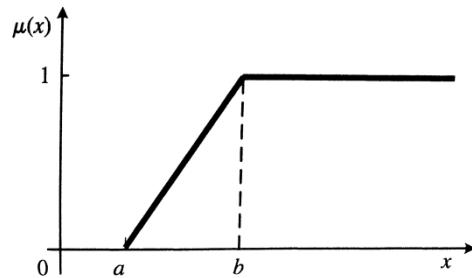
---

<sup>26</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Ilustracja 2.12. Funkcja przynależności klasy  $\pi$ .<sup>27</sup>

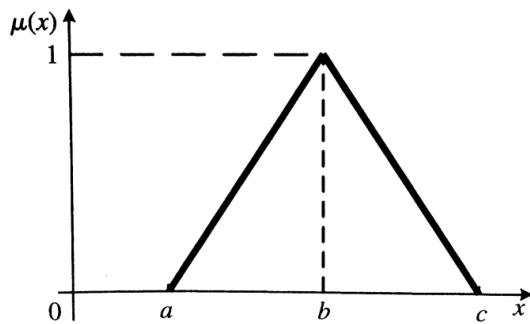
- Funkcja przynależności klasy  $\gamma$

$$\gamma(x; a, b) = \begin{cases} 0, & \text{dla } x \leq a, \\ \frac{x-a}{b-a}, & \text{dla } a < x \leq b, \\ 1, & \text{dla } x > b. \end{cases} \quad (13)$$

Ilustracja 2.13. Funkcja przynależności klasy  $\gamma$ .<sup>25</sup>

- Funkcja przynależności klasy t

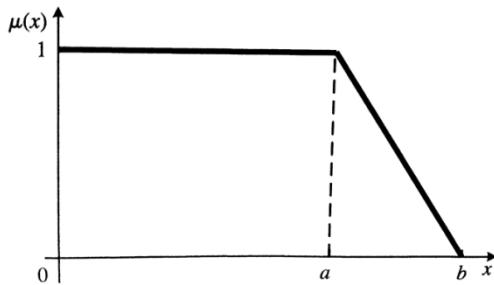
$$t(x; a, b, c) = \begin{cases} 0, & \text{dla } x \leq a, \\ \frac{x-a}{b-a}, & \text{dla } a < x \leq b, \\ \frac{c-x}{c-b}, & \text{dla } b < x \leq c, \\ 1, & \text{dla } x > c. \end{cases} \quad (14)$$

Ilustracja 2.14. Funkcja przynależności klasy  $t$ .<sup>25</sup>

<sup>27</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

- Funkcja przynależności klasy L

$$L(x; a, b) = \begin{cases} 0, & \text{dla } x \leq a, \\ \frac{b-x}{b-a}, & \text{dla } a < x \leq b, \\ 1, & \text{dla } x > b. \end{cases} \quad (15)$$



Ilustracja 2.15. Funkcja przynależności klasy L.<sup>28</sup>

#### ▪ Własności zbiorów rozmytych

Nośnik zbioru rozmytego jest to zbiór elementów przestrzeni  $X$ , których funkcja przynależności  $\mu_A(x) > 0$ , dla pewnego zbioru rozmytego  $A$  i oznacza się  $\text{supp } A$  (ang. support):

$$\text{supp } A = \{x \in X; \mu_A(x) > 0\}. \quad (16)$$

Dla przykładu<sup>29</sup>, jeżeli  $X$  jest przestrzenią dyskretną taką, że  $X = \{1,2,3,4,5,6\}$  oraz określono zbiór rozmyty na tej przestrzeni  $A = \frac{0,2}{1} + \frac{0,4}{2} + \frac{0,7}{4}$  to  $\text{supp } A = \{1,2,4\}$ .

Wysokość zbioru rozmytego dla pewnego zbioru rozmytego  $A$  oznacza się  $h(A)$  i zapisuje:

$$h(A) = \sup_{x \in X} \mu_A(x). \quad (17)$$

Jeżeli  $X$  jest przestrzenią dyskretną taką, że  $X = \{1,2,3,4,5\}$  oraz określono zbiór rozmyty na tej przestrzeni  $A = \frac{0,3}{2} + \frac{0,8}{3} + \frac{0,5}{4}$  to  $h(A) = 0,8$ .

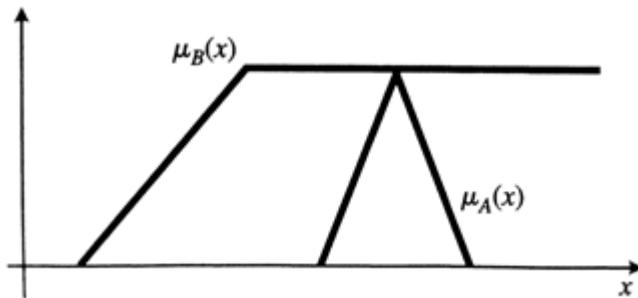
Zbiór rozmyty  $A$  nazywamy normalnym zbiorem rozmytym, wtedy i tylko wtedy, gdy  $h(A) = 1$ . Jeżeli tak nie jest, to zbiór ten można znormalizować za pomocą przekształcenia:

$$\mu_{A_{zn}}(x) = \frac{\mu_A(x)}{h(A)}. \quad (18)$$

<sup>28</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

<sup>29</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Zbiór rozmyty  $A$ , nazywamy pustym zbiorem rozmytym, wtedy i tylko wtedy, gdy  $\mu_A(x) = 0$  dla każdego  $x \in X$ . Zapisujemy go jako  $A = \emptyset$ . Zawieranie zbiorów występuje, gdy zbiór rozmyty  $A$  zawiera się w zbiorze rozmytym  $B$ , co zapisujemy  $A \subset B$ , wtedy i tylko wtedy, gdy  $\mu_A(x) \leq \mu_B(x)$  dla każdego  $x \in X$ .



Ilustracja 2.16. Zbiór B zawiera zbiór A.<sup>30</sup>

Równość zbiorów występuje wtedy, gdy zbiór rozmyty  $A$  jest równy zbiorowi rozmytemu  $B$ , co zapisujemy  $A = B$ , wtedy i tylko wtedy, gdy  $\mu_A(x) \leq \mu_B(x)$  dla każdego  $x \in X$ .

Przekrojem  $\alpha$  zbioru rozmytego  $A \subseteq X$ , nazywamy zbiór nierozmyty  $A_\alpha$ , określony jako:

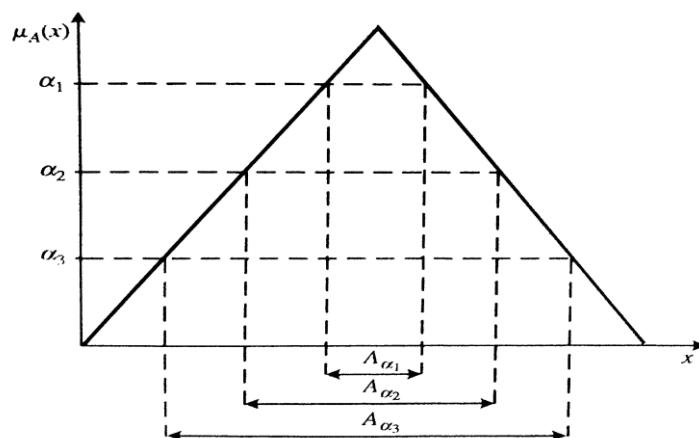
$$A_\alpha = \{x \in X; \mu_A(x) > \alpha\}, \forall_{\alpha \in [0,1]}, \quad (19)$$

Zbiór ten określony jest przez funkcję charakterystyczną:

$$\chi_{A_\alpha}(x) = \begin{cases} 1 & \text{dla } \mu_A(x) \geq \alpha, \\ 0 & \text{dla } \mu_A(x) < \alpha. \end{cases} \quad (20)$$

Można zauważyc, że zachodzi następująca implikacja:

$$\alpha_2 < \alpha_1 \Rightarrow A_{\alpha_1} \subset A_{\alpha_2}. \quad (21)$$



Ilustracja 2.17. Przekroje  $\alpha$  zbioru rozmytego  $A$ .<sup>27</sup>

<sup>30</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Dla dyskretnego zbioru rozmytego  $A \subseteq X$ , gdzie  $X = [1, \dots, 10]$ :<sup>31</sup>

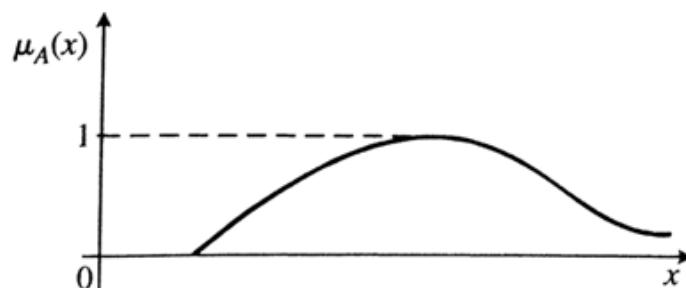
$$A = \frac{0,1}{2} + \frac{0,3}{4} + \frac{0,7}{5} + \frac{0,8}{8} + \frac{1}{10} \quad (22)$$

Zgodnie z podaną definicją  $\alpha$ -przekroje określone są następujące:

$$\begin{aligned} A_0 &= X = [1, \dots, 10], \\ A_{0,1} &= [2, 4, 5, 8, 10], \\ A_{0,3} &= [4, 5, 8, 10], \\ A_{0,7} &= [5, 8, 10], \\ A_{0,8} &= [8, 10], \\ A_1 &= [10]. \end{aligned} \quad (23)$$

Zbiór rozmyty nazywamy zbiorem rozmytym wypukłym  $A \subseteq R$ , wtedy i tylko wtedy, gdy dla dowolnego  $x_1, x_2 \in R$  i  $\lambda \in [0,1]$  zachodzi:

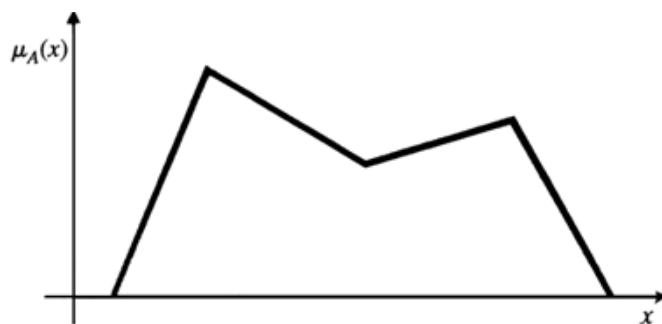
$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min\{\mu_A(x_1), \mu_A(x_2)\}. \quad (24)$$



Ilustracja 2.18. Zbiór rozmyty wypukły.<sup>27</sup>

Zbiór rozmyty nazywamy zbiorem rozmytym wklęsłym  $A \subseteq R$ , wtedy i tylko wtedy, gdy istnieją takie punkty  $x_1, x_2 \in R$  i  $\lambda \in [0,1]$ , że spełniona jest nierówność:

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] < \min\{\mu_A(x_1), \mu_A(x_2)\}. \quad (25)$$



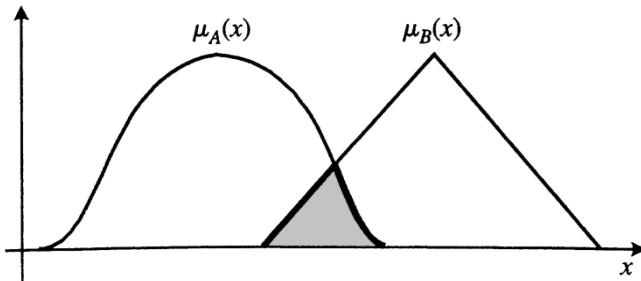
Ilustracja 2.19. Zbiór rozmyty wklęsły.<sup>27</sup>

<sup>31</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

▪ **Operacje na zbiorach rozmytych**

„Przecięciem zbiorów rozmytych  $A, B \subseteq X$  jest zbiór rozmyty  $A \cap B$  o funkcji przynależności” [13]:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad (26)$$



Ilustracja 2.20. Graficzne przedstawienie działania operacji przecięcia zbiorów rozmytych.<sup>32</sup>

Należy również podkreślić, że nie jest to jedyna definicja przecięcia zbiorów rozmytych. Ogólniej można je zdefiniować jako:

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) \quad (27)$$

w tym przypadku:

$$\min(\mu_A(x), \mu_B(x)) = T(\mu_A(x), \mu_B(x)) \quad (28)$$

natomiast przecięciem n zbiorów rozmytych  $A_1, A_2, \dots, A_n \subseteq X$  jest zbiór rozmyty  $A = A_1 \cap \dots \cap A_n = \bigcap_{i=1}^n A_i$  określony przez funkcję przynależności:

$$\mu_A(x) = T_{i=1}^n \mu_{A_i}(x). \quad (29)$$

Funkcja T jest tzw. t-normą, określającą operację przecięcia zbiorów rozmytych. Funkcja ta posiada kilka definicji, na potrzeby tej pracy wykorzystywane będą tylko jej dwa rodzaje. Pierwszą z nich jest funkcja minimum dana wzorem (26), natomiast druga ma postać iloczynu algebraicznego (36):

$$\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x) \quad (30)$$

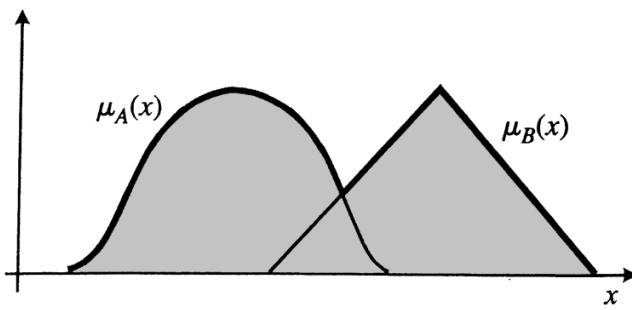
lub ogólniej:

$$\mu_A(x) = \prod_{i=1}^n \mu_{A_i}(x) \quad (31)$$

Sumą zbiorów rozmytych A i B jest zbiór rozmyty  $A \cup B$  o funkcji przynależności:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad (32)$$

<sup>32</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]



Ilustracja 2.21. Graficzne przedstawienie działania operacji sumy zbiorów rozmytych.<sup>33</sup>

Również w tym przypadku, nie jest to jedyna definicja sumy zbiorów, ogólniej sumę n zbiorów rozmytych  $A_1, A_2, \dots, A_n \subseteq X$  można zdefiniować jako zbiór  $A = A_1 \cup \dots \cup A_n = \bigcup_{i=1}^n A_i$  określony przez funkcję przynależności:

$$\mu_A(x) = S_{i=1}^n \mu_{A_i}(x). \quad (33)$$

Funkcja S jest tzw. t-konormą i określa operację sumy zbiorów:

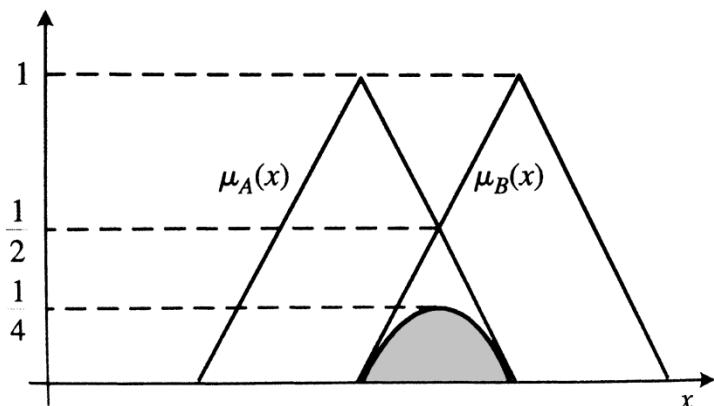
$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) \quad (34)$$

w tym przypadku

$$\max(\mu_A(x), \mu_B(x)) = S(\mu_A(x), \mu_B(x)) \quad (35)$$

Do dalszej części pracy nie jest wymagana znajomość innych przykładów funkcji t-konormy. Iloczynem algebraicznym zbiorów A i B jest zbiór rozmyty  $C = A \cdot B$  zdefiniowany funkcją przynależności:

$$C = \{(x, \mu_A(x) \cdot \mu_B(x)) | x \in X\}. \quad (36)$$



Ilustracja 2.22. Graficzne przedstawienie działania operacji iloczynu algebraicznego zbiorów rozmytych.<sup>29</sup>

<sup>33</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Każdy zbiór rozmyty  $A \subseteq X$  można przedstawić w postaci dekompozycji zbioru rozmytego:

$$A = \bigcup_{\alpha \in [0,1]} \alpha A_\alpha, \quad (37)$$

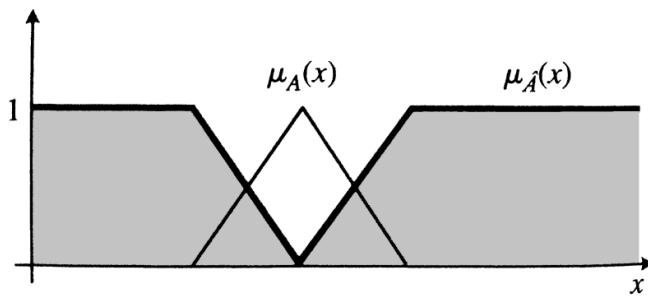
gdzie  $\alpha A_\alpha$  oznacza zbiór rozmyty, którego elementom przypisano następujące stopnie przynależności:

$$\mu_{\alpha A_\alpha}(x) = \begin{cases} \alpha & \text{dla } x \in A_\alpha, \\ 0 & \text{dla } x \notin A_\alpha. \end{cases} \quad (38)$$

Dopełnieniem zbioru rozmytego  $A \subseteq X$  jest zbiór rozmyty  $\hat{A}$  o funkcji przynależności:

$$\mu_{\hat{A}}(x) = 1 - \mu_A(x) \quad (39)$$

dla każdego  $x \in X$ .



Ilustracja 2.23. Graficzne przedstawienie działania operacji dopełnienia zbioru rozmytego.<sup>34</sup>

Iloczyn kartezjański zbiorów rozmytych  $A \subseteq X$  i  $B \subseteq Y$  oznacza się  $A \times B$  i definiuje jako:

$$\mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y)) \quad (40)$$

lub

$$\mu_{A \times B}(x, y) = \mu_A(x)\mu_B(y) \quad (41)$$

dla każdego  $x \in X$  i  $y \in Y$ . Iloczyn kartezjański n zbiorów rozmytych  $A_1 \subseteq X_1, A_2 \subseteq X_1, \dots, A_n \subseteq X_n$  oznaczamy  $A_1 \times A_2 \times \dots \times A_n$ , jest określony przez:

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \prod_{i=1}^n \mu_{A_i}(x_i). \quad (42)$$

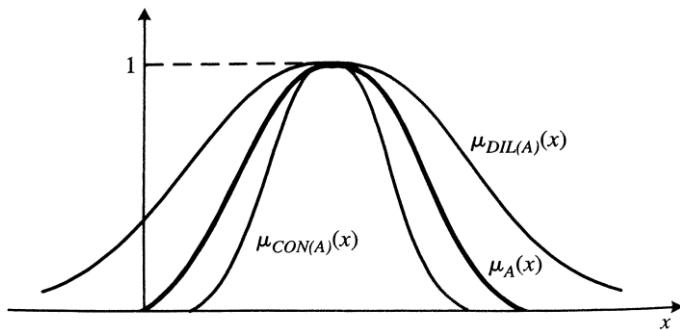
Koncentracja zbioru rozmytego  $A \subseteq X$  jest oznaczana  $CON(A)$  i zdefiniowana jako

$$\mu_{CON(A)}(x) = (\mu_A(x))^2 \quad (43)$$

dla każdego  $x \in X$ . Lingwistycznie koncentracją zbioru oznacza "bardzo" np. "bardzo stary" dla zdefiniowanego zbioru "stary". Rozcieńczenie zbioru rozmytego  $A \subseteq X$ , dla każdego  $x \in X$  jest oznaczane  $DIL(A)$  i definiowane jako:

$$\mu_{DIL(A)}(x) = (\mu_A(x))^{0,5} \quad (44)$$

<sup>34</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]



Ilustracja 2.24. Przedstawienie operacji koncentracji oraz rozcieńczania zbioru rozmytego.<sup>35</sup>

Lingwistycznie rozcieńczenie oznacza zbioru oznacza "mniej więcej" lub "około" np. "mniej więcej bogaty".

#### ■ Relacja Rozmyta

Pojęcie relacji rozmytej jest jednym z fundamentalnych w teorii zbiorów rozmytych, pozwala ona sformalizować nieprecyzyjne stwierdzenia typu "*x jest prawie równe y*" lub "*x jest znacznie mniejsze od y*" [13].

„*Relację rozmytą R między dwoma niepustymi zbiorami (nieroźmytymi) X i Y jest zbiór rozmyty określony na iloczynie kartezjańskim X × Y*” [13], tzn.

$$R \subseteq X \times Y = \{(x, y) : x \in X, y \in Y\}. \quad (45)$$

Ujmując to inaczej, relacja rozmyta jest zbiorem par:

$$R = \{((x, y), \mu_R(x, y))\}, \forall_{x \in X}, \forall_{y \in Y}, \quad (46)$$

gdzie:

$$\mu_R : X \times Y \rightarrow [0, 1] \quad (47)$$

jest funkcją przynależności. „*Funkcja ta każdej parze (x, y), x ∈ X, y ∈ Y przypisuje jej stopień przynależności μ\_R(x, y), który interpretuje się jako siłę powiązania między elementami x ∈ X i y ∈ Y*” [13]. Relację rozmytą można zapisać w postaci:

$$R = \sum_{X \times Y} \frac{\mu_R(x, y)}{(x, y)} \quad (48)$$

dla zbioru dyskretnego lub dla zbioru ciągłego:

$$R = \int_{X \times Y} \frac{\mu_R(x, y)}{(x, y)} \quad (49)$$

<sup>35</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

Dla przykładu<sup>36</sup> sformalizowane zostanie nieprecyzyjne stwierdzenie "y jest mniej więcej równe x". Niech  $X = \{3,4,5\}$  oraz  $Y = \{4,5,6\}$ . Relację R przykładowo można zdefiniować jako:

$$R = \frac{1}{(4,4)} + \frac{1}{(5,5)} + \frac{0,8}{(3,4)} + \frac{0,8}{(4,5)} + \frac{0,8}{(5,4)} + \frac{0,8}{(5,6)} + \frac{0,6}{(3,5)} + \frac{0,6}{(4,6)} + \frac{0,4}{(3,6)}. \quad (50)$$

Funkcja przynależności  $\mu_R(x, y)$  w tym przypadku przyjmie postać:

$$\mu_R(x, y) = \begin{cases} 1, & \text{jeżeli } x = y \\ 0,8, & \text{jeżeli } |x - y| = 1 \\ 0,6, & \text{jeżeli } |x - y| = 2 \\ 0,4, & \text{jeżeli } |x - y| = 3 \end{cases} \quad (51)$$

Relację R można również zapisać w postaci macierzy:

$$R = \begin{matrix} & y_1 & y_2 & y_3 \\ \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} & \begin{bmatrix} 0,8 & 0,6 & 0,4 \\ 1 & 0,8 & 0,6 \\ 0,6 & 1 & 0,8 \end{bmatrix} \end{matrix} \quad (52)$$

gdzie  $x_1 = 3, x_2 = 4, x_3 = 5$  oraz  $y_1 = 4, y_2 = 5, y_3 = 6$ .

„Złożeniem zbioru rozmytego  $A \subseteq X$  i relacji rozmytej  $R \subseteq X \times Y$  o oznaczeniu  $A \circ R$  nazywamy zbiór rozmyty  $B \subseteq Y$ ” [13]:

$$B = A \circ R \quad (53)$$

o funkcji przynależności:

$$\mu_B(y) = \sup_{x \in X} \{\mu_A(x)^T * \mu_R(x, y)\}. \quad (54)$$

Postać tego wzoru zależy od przyjętej t-normy (30) oraz od właściwości zbioru  $X$ . Wyróżnione zostaną 4 przypadki [13]:

1. Jeżeli  $T(a, b) = \min[a, b]$ , to powstanie złożenie typu sup-min

$$\mu_B(y) = \sup_{x \in X} \{\min[\mu_A(x), \mu_R(x, y)]\}. \quad (55)$$

2. Jeżeli  $T(a, b) = \min[a, b]$  oraz  $X$  jest zbiorem o skończonej liczbie elementów, to powstanie złożenie typu max-min

$$\mu_B(y) = \max_{x \in X} \{\min[\mu_A(x), \mu_R(x, y)]\}. \quad (56)$$

3. Jeżeli  $T(a, b) = a \cdot b$ , to powstanie złożenie typu sup-iloczyn

$$\mu_B(y) = \sup_{x \in X} \{\mu_A(x) \cdot \mu_R(x, y)\}. \quad (57)$$

---

<sup>36</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

4. Jeżeli  $T(a, b) = a \cdot b$  oraz  $X$  jest zbiorem o skończonej liczbie elementów, to powstanie złożenie typu max-iloczyn:

$$\mu_B(y) = \max_{x \in X} \{\mu_A(x) \cdot \mu_R(x, y)\}. \quad (58)$$

Dla przykładu<sup>37</sup>, niech przestrzenie  $X$  i  $Y$  będą określone jako  $X = \{x_1, x_2, x_3\}$  oraz  $Y = \{y_1, y_2, y_3\}$ . Zbiór rozmyty A jest zdefiniowany jako.

$$A = \frac{0,4}{x_1} + \frac{1}{x_2} + \frac{0,6}{x_3}, \quad (59)$$

Relację  $R$  reprezentuje macierz

$$R = \begin{matrix} & y_1 & y_2 \\ x_1 & 0,5 & 0,7 \\ x_2 & 0,2 & 1 \\ x_3 & 0,9 & 0,3 \end{matrix} \quad (60)$$

Wyznaczone zostanie złożenie  $A \circ R$  typu max-min zgodnie ze wzorem (56). Rezultatem złożenia jest zbiór rozmyty B postaci:

$$B = \frac{\mu_B(y_1)}{y_1} + \frac{\mu_B(y_2)}{y_2}, \quad (61)$$

gdzie:

$$\begin{aligned} \mu_B(y_1) &= \max\{\min(0,4; 0,5), \min(1; 0,2), \min(0,6; 0,9)\} = 0,6, \\ \mu_B(y_2) &= \max\{\min(0,4; 0,7), \min(1; 1), \min(0,6; 0,3)\} = 1, \end{aligned}$$

W rezultacie:

$$B = \frac{0,6}{y_1} + \frac{1}{y_2}. \quad (62)$$

---

<sup>37</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji” [13]

### 3. Sztuczna Inteligencja w grach

*"Inteligencja jest czymś czego używasz gdy nie wiesz co zrobić."*

Jean Piaget<sup>38</sup>

Celem stawianym przed twórcami systemów Sztucznej Inteligencji na potrzeby gier nie jest tworzenie systemów naśladowujących ludzki proces myślowy, lecz stworzenie pewnej imitacji inteligencji. Jak stwierdza B. Schwab w swojej książce [14, s. 2] "Sztuczna Inteligencja w grach jest głównie kodem w grze, który sprawia, że elementy sterowane przez nią wydają się podejmować mądre decyzje w sytuacjach gdzie mają wybór wynikające z tego zachowanie jest odpowiednim, skutecznym oraz użytecznym."<sup>39</sup> Dlatego też w grach komputerowych, twórcy nie skupią się na procesie myślowym lecz na racjonalnym działaniu agentów, aby postacie w efekcie robiły wrażenie intelligentnych, co nie zawsze zmusza projektantów do stosowania wymyślnych technik, ponieważ czasem wystarczą pewne sztuczki, by gracz postrzegał zachowanie postaci w grze jako intelligentne. Przykładem tutaj jest gra Halo, której projektanci odkryli, że testerzy gry mogli być bardzo łatwo oszukani myśląc, że przeciwnicy są bardziej intelligentni, poprzez zwyczajne zwiększenie liczby punktów życia przeciwników. Zabieg ten zwiększał tylko ich wytrzymałość, przez co również trudność w ich pokonaniu<sup>40</sup>.

Sztuczną Inteligencję w grach komputerowych można przydzielić do kategorii systemów zachowujących się jak ludzie<sup>41</sup>. Zachowujących się ponieważ to co jest ważne w grach to efekt działania jednostek, a nie ich proces myślowy. Jak ludzie ponieważ gry mają dawać rozrywkę graczom, którzy powinni być lepsi, dla tego system powinien popełniać błędy. [14]

---

<sup>38</sup> Jean Piaget (1896 - 1980) - psycholog oraz filozof, cytat pochodzi z [15], tłumaczenie własne.

<sup>39</sup> Tłumaczenie własne.

<sup>40</sup> Przykład z książki "Programming Game AI by Example" [16, s. 22]

<sup>41</sup> Jedna z kategorii definiujących Sztuczną Inteligencję przedstawiona przez autorów S. Russell, P. Norvig w książce „Artificial Intelligence: A Modern Approach” [3]

### 3.1 Krótka Historia

"Jeśli się nie zna historii, pozostaje się zawsze dzieckiem."

Cyberon<sup>42</sup>

W początkowej fazie rozwoju gier komputerowych producenci przywiązywali głównie wagę do grafiki, jednym z powodów były ograniczenia co do czasu pracy procesora, a ponieważ grafika robiła największe wrażenie na gracach, Sztuczna Inteligencja została zepchana na drugi plan i polegała głównie na sztywno zakodowanych wzorcach zachowania lub na prostych maszynach stanów [22]. Obecnie większość algorytmów oraz obliczeń związanych z grafiką jest wykonywana poprzez układy graficzne komputerów, a polepszająca się grafika w grach już nie robi takiego wrażenia na gracach, którzy wymagają czegoś więcej od gier. Dlatego też producenci kładą coraz to większy nacisk na rozwój Sztucznej Inteligencji, przy coraz to większych oczekiwaniach graczy, gdzie proste sztuczki już nie wystarczają, aby ich zadowolić [22].

Jedną z pierwszych gier wideo stworzył Williama Higinbothama w 1958 roku. Była to gra "*Tennis for Two*" będąca symulacją gry w tenisa ziemnego, gdzie do wyświetlenia obrazu wykorzystano oscyloskop [18], natomiast pierwszą grą wideo stworzoną z przeznaczeniem na komputer było "*Space War*", napisaną na Uniwersytecie Technicznym w Massachusetts przez S. Russel na minikomputer PDP1 w 1962 roku [19]. Gry te wymagały jednak dwóch graczy do rozgrywki. Dopiero lata siedemdziesiąte przyniosły pierwsze założki Sztucznej Inteligencji w grach, były to głównie obiekty poruszające się według pewnych prostych, ustalonych wzorców. Przykładem może być gra z roku 1978 znana jako "*Space Invaders*", polegająca na zniszczeniu wrogich jednostek, poruszających się w przestrzeni kosmicznej [22].

W 1979 roku powstał słynny „*Pac-Man*” [Midway Games West, 1979], pierwsza gra w której gracz naprawdę miał przeciwników, którzy jak na tamte czasy sprawiali wrażenie inteligentnych, posiadających własną "osobowość" o odmiennym zachowaniu<sup>43</sup> [14, s. 4]. Przeciwnicy poruszali się po labiryncie ścigając gracza, w sposób sprawiający wrażenie ich wzajemnej współpracy przeciwko niemu. Pac-Man bazował na prostej maszynie stanów, gdzie każdy z czterech duszków mógł gonić lub uciekać przed graczem po labiryncie [23, s. 7]. Póki co jednak gry bazowały na prostych lub bardziej zaawansowanych wzorcach, jak w klasycznych grach „*Golden Axe*” [SEGA Entertainment, 1987] czy „*Super Mario Brothers*” [Nintendo, 1985]. Ruchy

<sup>42</sup> Cyberon (106 - 43 p.n.e.) - mówca, filozof i polityk rzymski[17]

<sup>43</sup> Polegało to na tym, że podczas pościgu, na każdym z rozwidleń dróg duszki miały różną szansę wyboru losowej drogi lub pogoni za graczem [23, s. 7]

przeciwnika polegały głównie na poruszaniu się w jednym lub dwóch kierunkach, aż do napotkania gracza, który ma za zadanie go pokonać. [23, s. 7].

Gra „*Goldeneye 007*” [Rare Ltd., 1997] była pierwszą grą FPS (ang. First Person Shooter)<sup>44</sup> posiadającą Sztuczną Inteligencję potrafiącą zareagować odpowiednio na ruch oraz akcję gracza [21]. Postacie w grze posiadały zmysł wzroku, były w stanie spostrzec gdy ich koledzy lub partnerzy byli martwi [23, s. 8]. W grze „*Thief: The Dark Project*” [Looking Glass Studios, 1998] rozgrywka już w całości opierała się na symulacji zmysłów wzroku oraz słuchu<sup>45</sup>.

Znaczenie AI w grach zaczynało rosnąć, gdy gry takie jak „*The Sims*” [Maxis Software, 2000], gdzie wymodelowano ludzkie emocje i potrzeby oraz „*Black and White*” [Lionhead Studios Ltd., 2001], gdzie agent z umysłem bazującym na sztucznej sieci neuronowej może się uczyć w sztucznie stworzonym środowisku, sprawiło, że ludzie z niedowierzaniem patrzyli na poziom AI. Jednak ciągle dużo gier korzysta z podstawowych technik w tworzeniu Sztucznej Inteligencji, często koncentrujących się tylko na prostych wzorcach, ponieważ nie potrzebują niczego więcej. Przykładem mogą być gry na konsole lub telefony komórkowe. Wciąż najczęściej używaną techniką są maszyny stanów oraz jej pochodne.

### 3.2 Inteligentny agent bazujący na logice rozmytej

Omówiony zostanie model sztucznej inteligencji w grze, która była inspiracją napisania tej pracy. Analizie zostanie poddana gra „*Battle City*”. Zademonstrowane zostaną interesujące rozwiązania zastosowania inteligentnych agentów bazujących na logice rozmytej. Rozpatrzenie tego modelu umożliwi nam jego zróżnicowanie pod względem złożoności i sprawdzenie jego funkcjonowania w określonych warunkach.

Gra „*Battle City*” jest zręcznościowa strzelanka, wydana w 1985 roku na konsole gier wideo firmy Nintendo<sup>46</sup> przez Namco<sup>47</sup>. Gracz, kontrolujący czołg musi wyeliminować nieprzyjacielskie jednostki, próbujące go zniszczyć, jak i jego bazę reprezentowaną przez sokoła, który w początkowej fazie gry znajduje się za bezpiecznym murem obronnym. Gra składa się z 35 poziomów, w tym każdy zawiera różne typy terenu i przeszkód. Między innymi są to: ceglane mury, które ulegają zburzeniu po jednym strzale gracza lub wrogiego czołgu; stalowe ściany, które mogą być zniszczone jedynie po ulepszeniu jednostki przez gracza do wyższego poziomu; krzaki, skrywające czołgi pod nimi; pokrywy lodowe, utrudniające graczowi poruszanie się po nich; rzeki,

<sup>44</sup> Gra opierająca się na strzelaniu do przeciwników z widoku pierwszej osoby.

<sup>45</sup> Szczegóły dotyczące gry zostaną przedstawione w kolejnym dziale.

<sup>46</sup> Światowy lider w tworzeniu interaktywnej rozrywki.

<sup>47</sup> Japońska korporacja, były producent gier video. [8]

których czołg nie może przekroczyć. W grze występuje wiele rodzajów wrogich jednostek, pełniących różne zadania. Posiadają one różne kształty, kolory, prędkość, tarczę i siłę. Na planszy gry obecną są również chwilowe bonusy, wspomagające gracza w boju. Gra kończy się kiedy baza gracza zostanie zniszczona, lub gracz straci wszystkie dostępne życia.<sup>48</sup> Ilustracja 3.1 przedstawia scenę z przebiegu gry.



Ilustracja 3.1. Battle City<sup>49</sup>

W grze „Battle City” wszystkie wroglie czołgi współdzielą stochastyczne zachowanie, przez co stają się łatwymi celami. Decyzja o wystrzelaniu oraz kierunku poruszania się są obierane losowo. W rozwijanej na uniwersytecie Alberta w Edmonton, Kanada, grze „BattleCity.net” [24], kontrola nad bazą i działaniem czołgów została przejęta przez inteligentne jednostki – wykluczona została interwencja człowieka w ich sterowaniu. W tym celu twórcy gry zastosowali model BDI (ang. Belief-Desire-Intention – Przekonanie-Pragnienie-Zamiar), definiuje on wewnętrzną strukturę agenta. Model ma swoje korzenie w filozoficznych pracach E. Bartmana [27] badających relacje intencji, z innymi stanami mentalnymi. Jak sama nazwa wskazuje, struktura agenta w modelu BDI posiada trzy fundamentalne cechy:

- „Przekonanie” – wiedzę agenta o otaczającym go środowisku i samym sobie,
- „Pragnienie” – opisującym cel agenta, będący stanem, który agent chce osiągnąć,
- „Zamiar” – jest zestawem działań, które agent wykonuje, aby osiągnąć cel.

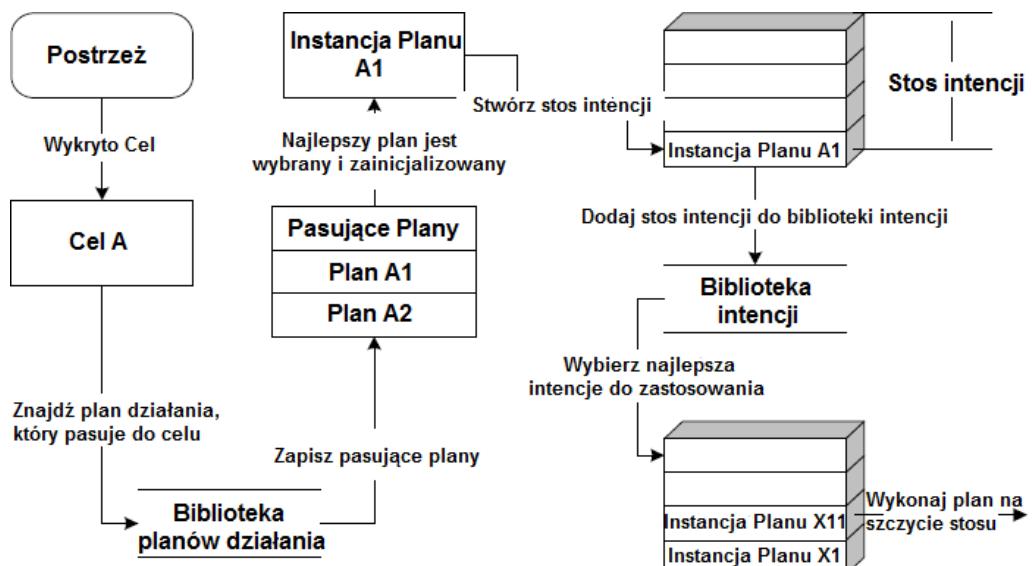
Istotnym elementem modelu BDI jest plan, będący przepisem działań, dzięki któremu agent nie musi przeszukiwać całej przestrzeni możliwych rozwiązań do osiągnięcia określonego celu. Agent postrzega otaczające go środowisko, aktualizując swój stan wiedzy. W procesie dążenia do osiągnięcia celu, przeszukiwana jest baza planów działania. Najlepszy z nich jest wybierany i zainicjalizowany. Tworzony jest stos „intencji”, z których to wykonywane są najbardziej pilne zestawy działań. W czasie wykonywania planu działania, do jego realizacji może wystąpić potrzeba osiągnięcia

<sup>48</sup> Opracowane na podstawie StrategyWiki, „BattleCity” [8].

<sup>49</sup> Jest to screenshot z gry.

<sup>49</sup> Screenshot (ang.) – zawartość ekranu w danym momencie, nazywana potocznie „zrzutem ekranu”.

innego, mniejszego celu, który jest poza zakresem możliwości aktualnego planu. W takiej sytuacji inicjalizowany jest nowy cel, którego proces wykonania jest identyczny do wyżej opisanego. W zależności od konieczności, nowe cele mogą zostać wykonywane w sposób synchroniczny. Ilustracja 3.2 przedstawia schemat jednego cyklu wykonawczego modelu BDI.net.<sup>50</sup>



Ilustracja 3.2. Cykl wykonawczy BDI.net<sup>51</sup>

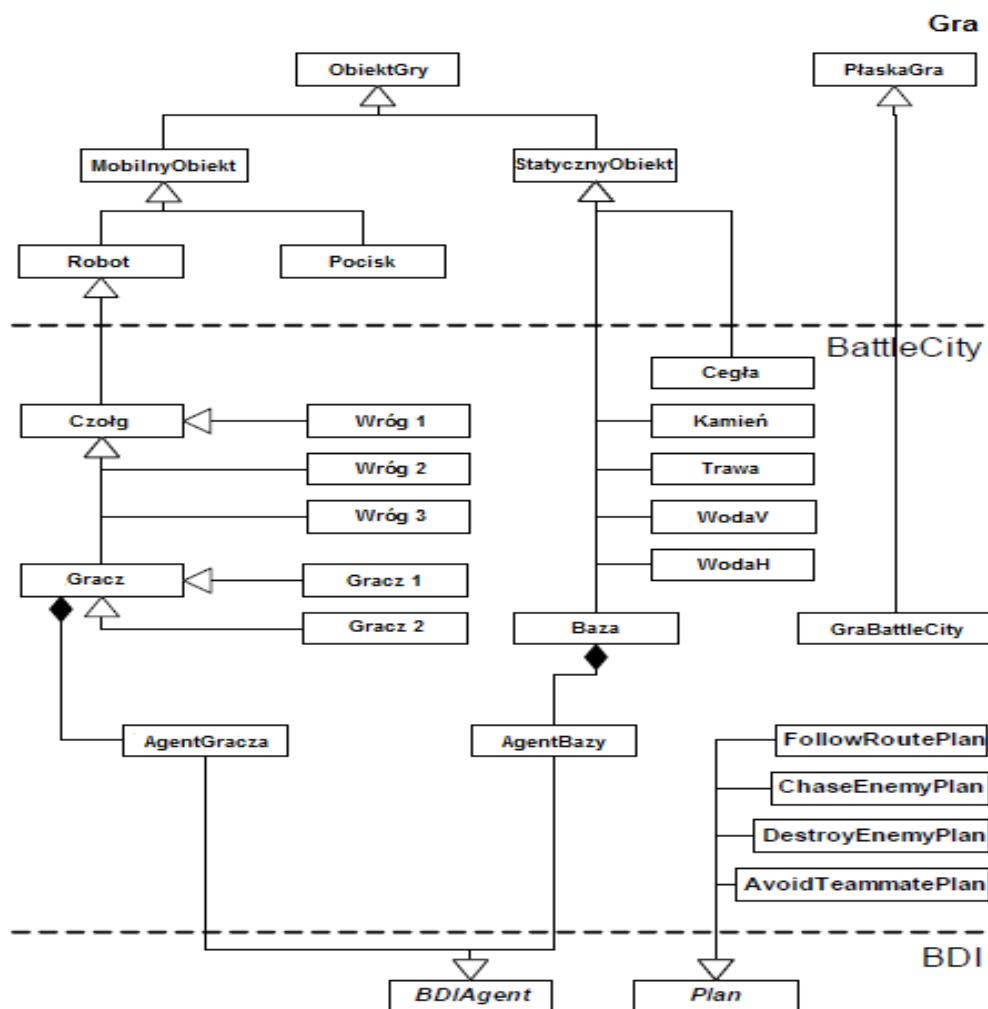
W grze zostało wprowadzone rozgraniczenie na obiekty nieożywione i mobilne jednostki. Czołg gracza i baza zostały zaprojektowane jako agenty, pozostawiając resztę jako zwykłe obiekty. Uproszczony diagram klas wybranych komponentów prezentuje Ilustracja 3.3. Rozdziela on strukturę gry na trzy główne elementy: moduł zarządzający, moduł implementujący oraz moduł BDI. Komunikacja w grze odbywa się na poziomie agentów – wymieniają oni wiadomości między sobą koordynując wspólną strategię. BDI.net implementuje język komunikacji agentów w standardzie FIPA (ang. Foundation for Intelligent Physical Agents). Specyfikuje on kodowanie wiadomości, semantykę oraz pragmatykę, wspiera XML (ang. Extensible Markup Language – Rozszerzalny Język Znaczników)<sup>52</sup>, specyfikuje protokoły interakcji. Przykładem komunikacji jest „wołanie o pomoc” przez bazę. W przypadku kiedy wroga jednostka zbliża się do bazy, wysyła ona wiadomość typu broadcast<sup>53</sup> do sprzymierzonych czołgów, które w odpowiedzi zwracają koszt udzielenia pomocy. Agent bazy nawiązuje kontakt z wybranym przez siebie czołgiem, na bieżąco informując go zagrożeniem. Zachowanie agenta generowane jest na

<sup>50</sup> Opracowana na podstawie rozdziału II A., B. z Y. Li, P. Musilek, L. Kurgan, „Battlecity Revived: Game Design with BDI.net” [26].

<sup>51</sup> Opracowana na podstawie ilustracji 1. z Y. Li, P. Musilek, L. Kurgan, „Battlecity Revived: Game Design with BDI.net” [26].

<sup>52</sup> XML jest uniwersalnym językiem do reprezentowania danych w strukturalnej formie.

<sup>53</sup> Broadcast – Rozgłoszeniowy tryb transmisji danych polegający na wysyłaniu jeden do wszystkich.

Ilustracja 3.3. Wybrane komponenty projektu „BattleCity.net”<sup>54</sup>

podstawie bazy reguł rozmytych, w której to zestaw instrukcji if-then-else<sup>55</sup> określa bodźce i odpowiadające im reakcje. System ten zwiększa poziom szczegółów procesu decyzyjnego. Przy ocenie zagrożenia agent bazy bierze pod uwagę czas od ostatniego ataku, poziom tarczy oraz połączone siły wrogów w zasięgu wzroku. Ilustracja 3.4 przedstawia te czynniki, posługując się wartościami zmiennych rozmytych. Na podstawie tych wejściowych czynników, baza reguł rozmytych zwraca zasięg, w którym mogą poruszać się sprzymierzone czołgi. Zaimplementowanych zostało 27 reguł, które zwracają dość zadawalające wyniki. Aby zredukować liczbę reguł i zapobiec jej wykładniczemu wzrostowi, zastosowana została konfiguracja URC (ang. Union Rule Configuration – Wspólna Konfiguracja Zasad). Zmniejsza ona trzykrotnie liczbę reguł, jak również polepsza wydajność gry. Typowa wspólna konfiguracja zasad wygląda następująco:

<sup>54</sup> Opracowana na podstawie ilustracji 3. z Y. Li, P. Musilek, L. Kurgan, „Battlecity Revived: Game Design with BDI.net” [26].

<sup>55</sup> Instrukcja warunkowa będąca elementarną cechą języków programowania, wykonująca działanie lub akcję na podstawie wejściowej wartości logicznej.

---

```

if  $X_1$  is  $A_{11}$  then  $Y$  is  $B_1$ 
OR if  $X_1$  is  $A_{12}$  then  $Y$  is  $B_2$ 
OR if  $X_2$  is  $A_{21}$  then  $Y$  is  $B_3$ 
OR ...
OR if  $X_i$  is  $A_{ij}$  then  $Y$  is  $B_k$ 

```

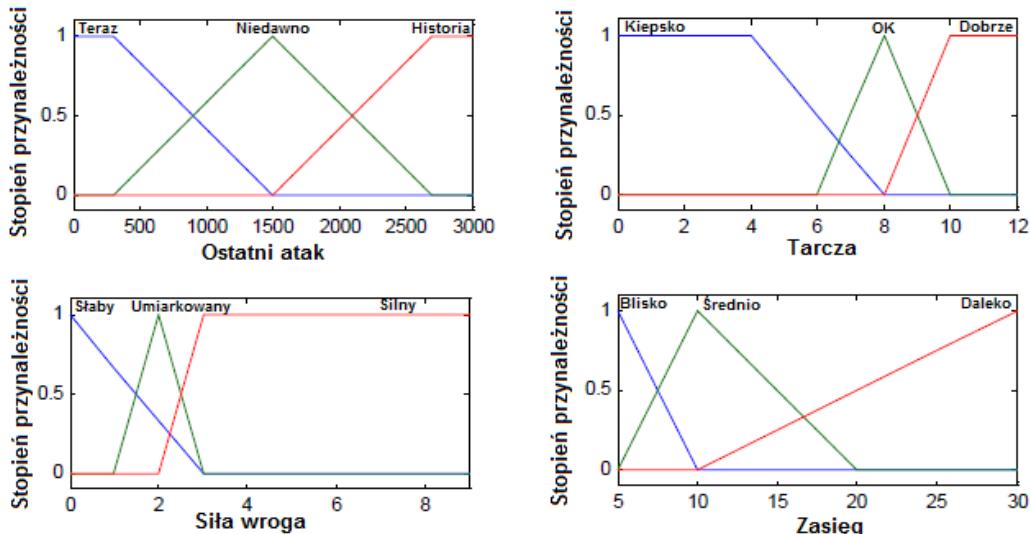
(63)

gdzie:

$X_i$  – oznacza zmienne wejściowe,

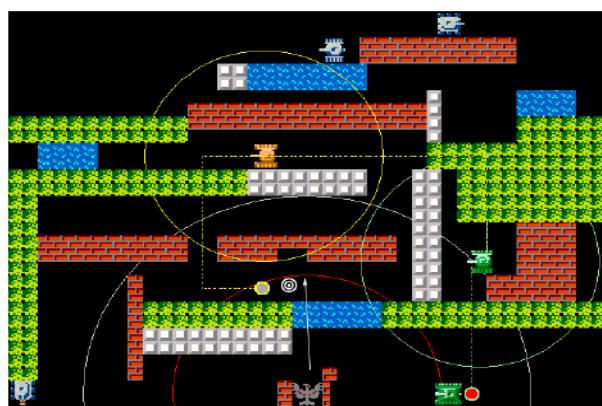
$A_{ij}$  i  $B_k$  – oznacza reguły językowe zdefiniowane w zmiennych rozmytych,

$Y$  – oznacza zmienne wyjściowe.



Ilustracja 3.4. Wartości zmiennych rozmytych do kontroli zasięgu czołgów<sup>56</sup>

Na początku gry wszystkie warunki są przychylne, tarcza bazy jest nie naruszona, żaden przeciwnik nie znajduje się w jej zasięgu oraz żaden atak nie został zarejestrowany. W rezultacie czołgi broniące bazy mogą niemalże swobodnie wędrować po mapie. Kiedy sytuacja się pogarsza, czołgi informowane są, aby zostały blisko bazy.



Ilustracja 3.5 Zrzut z ekranu z gry – „sytuacja awaryjna”<sup>57</sup>

---

<sup>56</sup> Opracowana na podstawie ilustracji 5. z Y. Li, P. Musilek, , L. Wyard-Scott, „Fuzzy Logic in Agent-Based Game Design” [24].

Ostatecznie, w sytuacji awaryjnej, które przedstawia Ilustracja 3.5, tarcza z prawej strony jest prawie całkowicie zniszczona i do bazy zbliża się wroga jednostka, co zmusza jeden z czołgów broniących do natychmiastowego pościgu atakującego napastnika. Drugi czołg, jak wynik procesu decyzyjnego, ma rozkaz pośpiesznego powrotu do bazy i jej obrony.

Podejście wdrażające agentów w tworzeniu gier i symulacji dostarcza naturalny sposób na modelowanie inteligentnych jednostek oraz wysoko elastycznej architektury. Umożliwia ona implementacje z rozmachem różnorodnych zachowań i łatwą integrację nowoczesnych projektów ze starszymi rozwiązaniami. Nie jest ono jednak bez wad. Problemy mogą pojawić się, gdy z braku zcentralizowanego planu działania i kontroli, autonomiczna natura agenta mogłaby doprowadzić do niepożądanych, nowo powstałych zachowań. Oczywiście każdy z takich przypadków mógłby być rozwiązywany z osobna. Twórcy gry „*BattleCity.net*” jednakże sugerują bardziej rozsądne podejście – stworzenie hybrydowej architektury, która łączy zcentralizowaną kontrolę oraz autonomicznego agenta z możliwością zewnętrznzej kontroli poprzez tzw. „tylne drzwi”. Logika rozmyta jest kolejną inteligentną techniką mogącą zwiększyć wydajność gier i symulacji. Jest w stanie obsługiwać złożony mechanizm kontroli przy minimalnych kosztach obliczeniowych, bez poświęcenia subtelnych detali. W tym przykładzie logika rozmyta została zastosowana w grze bazującej na inteligentnych agentach, naturalnie ma ona swoje zastosowanie w wielu zróżnicowanych architekturach.

### 3.3 Przykłady zastosowań AI w grach

W tym dziale chciałbym pokrótkę opisać kilka przełomowych gier, które głównie dzięki Sztucznej Inteligencji zdobyły dużą popularność, a niektórych kontynuacje są tworzone do dzisiaj. Opisane gry zostały wybrane na podstawie kilku czynników. Poniższe gry są nie najnowszymi produkcjami, ponieważ dostęp do bardziej szczegółowych informacji na ich temat jest ułatwiony. Sztuczna Inteligencja nie jest aż tak skomplikowanym systemem, jak to bywa w najnowszych produkcjach, skupia się głównie na kilku technikach. Pomimo tego, gry te znalazły swoje miejsce w rankingu najbardziej innowacyjnych gier w historii, umieszczonym na łamach serwisu „*AiGameDev*”<sup>57</sup>. W większości dzięki swojemu sukcesowi, zapoczątkowały serie gier, które wydawane są po dziś dzień. Dlatego też warto im się przyjrzeć trochę bliżej. Nie będziemy starali się tutaj potwierdzić kolejności w rankingu, które to gry były bardziej innowacyjne, a które mniej, jednak zgadzamy się z faktem, że wymienione tu gry były pionierami w swojej klasie oraz mierzyły się z wysokimi wyzwaniami stawianymi przed silnikiem Sztucznej Inteligencji z dużym sukcesem. Przedstawionych zostanie tutaj tylko

---

<sup>57</sup> Opracowana na podstawie ilustracji 8. z Y. Li, P. Musilek, , L. Wyard-Scott, „*Fuzzy Logic in Agent-Based Game Design*” [24].

<sup>58</sup> Serwis internetowy o tematyce sztucznej inteligencji w grach [22]

kilka gier z rankingu, wybranych głównie na podstawie dostępności materiałów opisujących nieco głębiej zaimplementowaną Sztuczną Inteligencję.

## ■ Thief

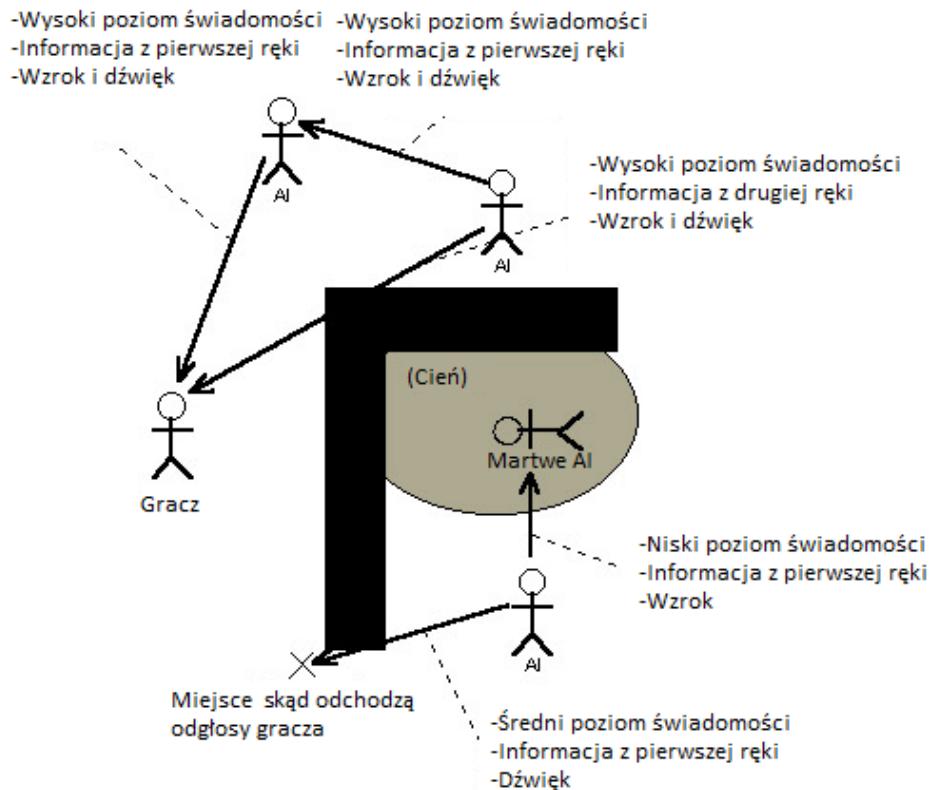
„*Thief: The Dark Project*” jest grą widzianą z perspektywy pierwszej osoby, opowiadającą o losach młodego złodzieja w czasach średniowiecznych. Gra nie jest typową strzelaniną, wręcz przeciwnie, w grze należy unikać przeciwników, gracz jest w pełni śmiertelny, a gra głównie polega na skradaniu się, unikaniu pułapek i wypełnianiu powierzonych złodziejowi misji. Sercem tego typu rozgrywki jest porządnego system czujników, poprzez które agenci w grze mogą odbierać otoczenie w podobny sposób jak odbiera je gracz. Jednym z głównych wymogów stawianych przed systemem, było operowanie na dużej liczbie stanów odczuwanych przez AI, a nie tylko czy gracz został dostrzeżony czy też nie. System powinien również działać w stosunku do obiektów innych niż gracz, np. zwłokach przeciwnika, które gracz powinien schować, by straż ich nie odnalazła. To wszystko powinno być dodatkowo zrozumiałe dla gracza, by był w stanie przewidzieć zachowanie przeciwników [28].



Ilustracja 3.6. Straż w pokoju w grze Thief. [22]

### Innowacje [22]:

- Gra korzysta z modelu czujników, pozwalając postaciom kontrolowanym przez AI na realistyczne reagowanie na bodźce dźwiękowe oraz świetlne,
- Postacie kontrolowane przez AI używają nagrań audio by oznać swój obecny stan, co pozwala graczowi na zorientowanie się w sytuacji.



Ilustracja 3.7. Przykład działania zmysłów w grze Thief<sup>59</sup>.

### Cechy Sztucznej Inteligencji [28], [29]:

Siła gry bazuje na systemie czujników dźwięku oraz wizji, jako osobnego modułu zbierającego informację z otoczenia w świecie gry. Na podstawie tak zebranych informacji postacie w grze mogą podejmować decyzje. Informacja jaką daje nam ten system to poziom świadomości AI, który jest wyrażony za pomocą zestawu dyskretnych wartości opisujących wiedzę AI na temat obecnej sytuacji, położenia oraz tożsamości obiektu zainteresowania. Widoczność w grze opisana jest przez światło, ruch, wielkość oraz widoczność na tle innych obiektów, natomiast sam zmysł wzroku AI jest opisany w trzech wymiarach. Na podstawie poziomu świadomości, AI określa odpowiednie działanie w procesie decyzyjnym. Sam proces decyzyjny opiera się na maszynie stanów.

### ■ The Sims

Symulator życia człowieka, a nawet całej rodziny oraz związane z tym obowiązki i problemy życia codziennego. Gracz staje przed wyzwaniem zaprojektowania i utrzymania gospodarstwa domowego wraz z jego mieszkańcami. Próba zaawansowanego zamodelowania agentów zbliżonych do ludzi, posiadających własną osobowość, potrzeby, umiejętności oraz relacje z innymi agentami. Wszystko to ma

<sup>59</sup> Ilustracja pochodzi z artykułu [29], tłumaczenie własne.

wpływ na ich zachowania, stosunek do innych, metody zaspokajania swoich potrzeb. Gra okazała się dużym sukcesem dającym początek całej serii.



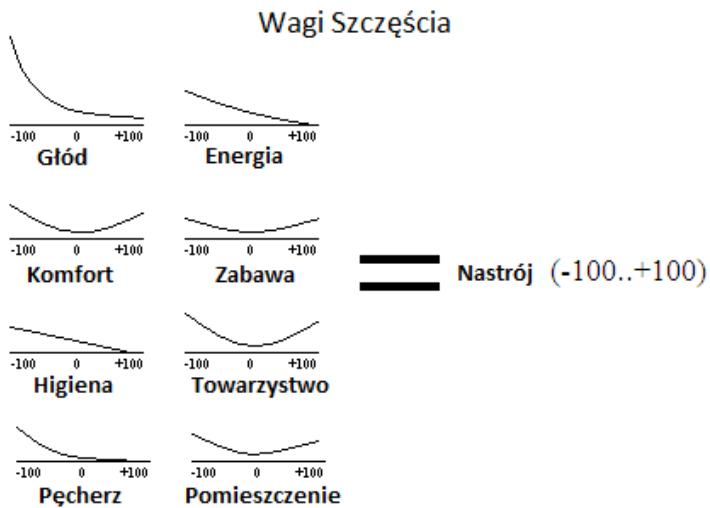
Ilustracja 3.8. Rodzina w domu w grze The Sims. [22]

### Innowacje [22]:

- Inteligentne obiekty zostały użyte by pomóc w implementacji zachowań. Obiekt definiuje w jaki sposób postać powinna wejść z nim w interakcję,
- Każda postać w grze ma swój charakter, umiejętności, podstawowe potrzeby fizyczne oraz emocjonalne, które kierują poczynaniami jednostki w grze. Pewna forma logiki rozmytej została tu wykorzystana do wymodelowania emocji postaci, które mierzone są w zakresie (-100, 100). Każda z tych potrzeb jest mapowana do wartości szczęścia/nastroju,
- Zamodelowano również emocjonalną interakcję pomiędzy osobami, co pozwala na tworzenie związków między nimi.

### Cechy Sztucznej Inteligencji:

Architektura personalna agenta w grze „*The Sims*” posiada cztery główne aspekty: osobowość, potrzeby, umiejętności oraz relacje z innymi agentami w grze [30]. Osobowość jest opisana wskaźnikiem określającym każdą z pięciu cech: niechlujny lub schludny, nieśmiały lub towarzyski, poważny lub żartobliwy, leniwy lub aktywny, złośliwy lub miły. Osobowość kieruje czynnościami postaci oraz określa jak na niego oddziaływają [30]. Potrzeby można podzielić na dwie grupy. Fizyczne: głód, komfort, higiena, pęcherz oraz umysłowe: energia, zabawa, towarzystwo, pomieszczenie. Każda z potrzeb jest opisana przez wartość w przedziale od -100 do 100. By pozyskać obecny nastrój agenta wszystkie potrzeby są modyfikowane poprzez odpowiadającą im wartość w funkcji wagi [30].



Ilustracja 3.9. Funkcje wag potrzeb agenta [30], wartości w zakresie (0,1)<sup>60</sup>.

Na przykład gdy higiena agenta jest na niskim poziomie, ten współczynnik ma bardzo wysoki wpływ na ogólny nastrój, lecz gdy jest na bardzo wysokim, wpływ na ogólny nastrój jest nieznaczny a nawet w skrajnym przypadku żaden. Ponieważ zakresy poszczególnych potrzeb oraz nastoju są takie same, wystarczy zsumować wszystkie czynniki by otrzymać poziom nastroju:

$$Nastrój = \frac{\sum_{i=1}^n \mu_i x_i}{\sum_{i=1}^n \mu_i} \quad 61 \quad (64)$$

gdzie:

$\mu_i$  - waga odpowiadająca potrzebie,

$x_i$  - wartość potrzeby.

W grze znajdują się tzw. intelligentne obiekty posiadające interfejs, który rozpowszechnia informacje do wszystkich agentów z listą możliwych do wykonania na nim akcji oraz w jaki sposób akcje te wpłyną na nich [31]. Na przykład założmy, że agent posiada następujące parametry (głód: +20, Komfort: -12, Higiena: -30, Pęcherz: -75, Energia: +80, Zabawa: +40, Towarzystwo: +10, Pomieszczenie: -60), co skutkuje ogólnym nastrojem na poziomie +18. Toaleta poinformowała agenta, że gdy z niej skorzysta, zaspokoi pęcherz oraz gdy ją wyczyści poprawi atmosferę w pokoju co skutkuje wzrostem jego nastroju do poziomu +26. Gdyby jednak użył wannę, poprawił by higienę oraz komfort co skutkowałoby wzrostem tylko do poziomu +20, ponieważ największy wpływ na jego nastrój ma pęcherz w procesie decyzyjnym jako następny obiekt do użycia wybierze toaletę [30]. System podejmuję decyzję na podstawie możliwych do

<sup>60</sup> Ilustracja z prezentacji [30], tłumaczenie własne, zakres wartości jest przypuszczeniem autorów.

<sup>61</sup> Przypuszczenia autorów na podstawie prezentacji [30].

wykonania akcji na danym obiekcie, wynagrodzenia jakie agent za nie otrzyma, odległości do obiektu oraz poziomu potrzeb. Jednak samo zachowanie agentów nie jest zbyt intelligentne, bazuje głównie na skryptach jakie udostępniają obiekty jak należy się z nimi obchodzić, jak np. skrypt odnoszący się do lodówki opisujący jak przygotować posiłek, jakie kolejne kroki należy wykonać [31].

## ■ Halo

„*Halo: Combat Evolved*” jest kolejną grą FPS w której gracz wciela się w rolę jednego z elitarnych żołnierzy i wyrusza na wojnę z obcą rasą. Gracz podczas rozgrywki może wykorzystać wiele elementów terenu takich jak zagłębenia, skały, drzewa, oraz może korzystać z przeróżnych pojazdów wodnych, lądowych czy powietrznych.

### **Innowacje [22]:**

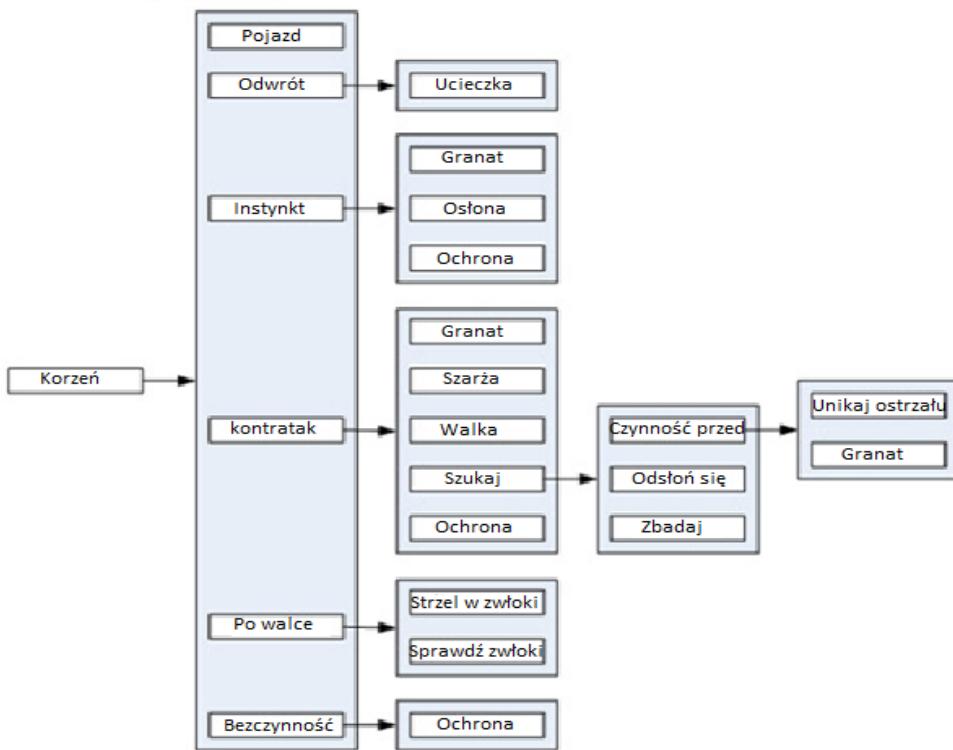
- Również mamy do czynienia z bardzo intelligentnymi jednostkami, potrafiącymi się kryć przed ostrzałem oraz używać rozważnie dostępnej broni,
- Wykorzystanie drzewa zachowań, które stało się bardzo popularne w przemyśle gier szczególnie od wyjścia "Halo 2",
- Sytuacja w jakiej znajduje się drużyna ma odzwierciedlenie w zachowaniu poszczególnych jednostek.

### **Cechy Sztucznej Inteligencji:**

U podstawy procesu decyzyjnego w grze leży architektura zwana drzewem zachowań<sup>62</sup> pozwalającą zapanować nad ich liczbą w grze. W architekturze tej odpowiedni węzeł wybierany jest na podstawie priorytetu oraz aktywności. Węzły potomne konkuruje ze sobą u rodzica o to, który zostanie wybrany. Proces ten odbywa się w kilku krokach, najpierw węzły dokonują pewnych obliczeń na podstawie zebranej wiedzy ze świata gry, co skutkuje tym czy są aktywne czy nie. Wybranym potomkiem będzie aktywny węzeł z najwyższym priorytetem, po zakończeniu działania następnego potomka w kolejności je zacznie. Jednak w czasie działania może nastąpić aktywacja potomka o wyższym priorytecie, który przerwie obecne działanie. Węzłami w drzewie mogą być pełnowartościowe zachowania jak walcz lub wejdź do pojazdu. Mogą nimi również być tzw. impulsy, które są tylko referencjami do zachowań, lecz tak jak one mogą być aktywne lub nie. Dzięki impulsom możliwe są działania w zależności od kontekstu, czasami zdarza się, że działanie A jest ważniejsze od działania B, a czasem nie.

---

<sup>62</sup> Tak nazwa została przyjęta[32], [33], jednak dokładnie jest to skierowany graf cykliczny.



Ilustracja 3.10. Przykład skierowanego grafu acyklicznego zachowań<sup>63</sup>.

Aktywność zachowań w drzewie jest sprawdzana bardzo często, czasami jednak przy zdarzeniach, które pojawiają się rzadko, niepotrzebnie tracony jest czas na testy aktywności. Na przykład mając impuls "Ucieczka gdy dowódca zginie", czeka on na wydarzenie „śmierć dowódcy w grze”. Testy jakie muszą zostać wykonane to między innymi, czy ten kto zginął był dowódcą, czy nie ma inny dowódców. Jeśli testy przejdą pozytywnie, referencja uruchamia zachowanie ucieczki. Takie testy wykonywane są co chwilę, by AI bazujące na drzewie mogło działać płynnie, pomimo tego, że zdarzenia takie jak opisane przed chwilą są bardzo rzadkie. By poradzić sobie z tym problemem twórcy wprowadzili system zdarzeń, który w odpowiednim momencie gry tymczasowo zaktualizuje strukturę drzewa by obsłużyć takie przypadki jak śmierć generała.

## ■ F.E.A.R

„F.E.A.R. First Encounter Assault” jest kolejną grą typu FPS, gdzie sterowany przez gracza bohater jest członkiem organizacji rządowej do spraw zwalczania terroryzmu. Podczas gry bohater wyrusza na jedną z misji w pięcioosobowej drużynie.

<sup>63</sup> Ilustracja na podstawie artykułu [33], tłumaczenie własne.

**Innowacje [22]:**

- Po raz pierwszy w głównym nurcie gier FPS użyto systemu planowania zadań do generowania zachowań zależnych od kontekstu. Ta technologia jest nadal używana przez studia jako punkt odniesienia,
- Przeciwnicy bardzo sprytnie wykorzystują otoczenie wirtualnego środowiska, znajdują osłonę, otwierają drzwi, przechodzą przez okna,
- Zaimplementowane zostały działania taktyczne drużyny między innymi: atak z flanki, czy używanie przerywanych serii ostrzału.

**Cechy Sztucznej Inteligencji [34]:**

Występują tylko dwa stany w niskopoziomowej maszynie stanów, które przyjmują wiele parametrów, przez co potrafią obsłużyć wiele zdarzeń. Użycie systemu planowania STRIPS (ang. Stanford Research Institute Problem Solver), który analizuje zależności każdej czynności i informuje w jaki sposób je zrealizować. Wykorzystanie architektury dynamicznej, czarnej tablicy, miejsca gdzie składowane są informacje o świecie w wyniku wydarzeń wokół AI.

W skrócie architektura agenta opiera się na czarnej tablicy, pamięci roboczej, czujnikach oraz kilku podsystemów odpowiedzialnych między innymi za wybór celu, nawigację, animację oraz broń. Zadaniem czujników jest wykrywanie zmian w otoczeniu, mogą one reagować na bodźce dźwiękowe oraz wizualne bazując na systemie zdarzeń lub pobierać dane bezpośrednio ze świata gry takie jak dostępność miejsc gdzie można się schronić przed ostrzałem. Tak pozyskane dane następnie są rejestrowane w pamięci roboczej. System planujący działanie agenta wykorzystuje je do podejmowania decyzji poprzez wysyłanie odpowiednich komunikatów do poszczególnych podsystemów, w tym celu wykorzystuje czarną tablicę.

**■ Black & White**

Seria „*Black & White*” [Lionhead Studios, 2001], jest grą, w której gracz wciela się w rolę boga zamieszkującego krainę różnych cywilizacji. Każda cywilizacja ma swoje przekonania, hobby, oraz moce. Celem gry jest przekonanie jak największej liczby mieszkańców wyspy do siebie jako ich boga. W grze znajdują się również inni bogowie, z którymi gracz musi konkurować o wpływ pośród cywilizacji. Walka pomiędzy bogami toczy się poprzez stworzenia, które są ich slugami. Gracz na początku gry dostaje jedno ze stworzeń do wyboru. Sercem samej gry jest właśnie to stworzenie, które sterowane przez Sztuczną Inteligencję musi zostać wyszkolone przez gracza. Uczy się jak należy się zachowywać poprzez obserwowanie gracza, obserwowanie innych lub poprzez stosowanie przez gracza kar i nagród. [18]



Ilustracja 3.11. Jedno ze stworzeń rzucające zaklęcie w grze Black & White. [22]

#### Innowacje [22]:

- Gra skupia się na interakcji gracza ze stworzeniem sterowanym przez AI, które może się uczyć w trakcie gry bazując na przykładach lub otrzymywaniu repremendy;
- Projekt integruje sztuczne życie w kontekście gry strategicznej. Drugim typem agenta w grze są jej mieszkańców, którzy pracują na rzecz wioski by ją rozwijać;
- Silnik wykorzystuje architekturę BDI (ang. Belief-Desire-Intention - Przekonanie-Pragnienie-Zamiar);
- Wykorzystanie technik z dziedziny nauki maszyn, takich jak drzewa decyzyjne oraz sztuczne sieci neuronowe.

#### Cechy Sztucznej Inteligencji [18]:

Użyto symbolicznych par parametr-wartość do opisu przekonań agenta na temat poszczególnych obiektów:

Wielkość przeszkód podczas poruszania:

- objekt.zbudowany-przez-człowieka.płot -> 1.0,
- objekt.natura.woda.płytko-rzeka -> 0.5,
- objekt.natura.kamień > 0.1

Ta metoda została użyta wraz z systemem bazującym na regułach by opisać podstawową wiedzę na temat obiektów znajdujących się w świecie. Jest to rodzaj AI bazującego na skryptach bardzo popularny w grach.



Ilustracja 3.12. Drzewo decyzyjne powstałe na podstawie powyższej tabeli<sup>64</sup>.

Drzewa decyzyjne reprezentują przekonania agenta na temat ogólnych typów obiektów, natomiast sieci neuronowe reprezentują jego pragnienia. Agent w grze może uczyć się na parę różnych sposobów, poprzez obserwowanie gracza, mieszkańców wyspy lub inny stworzeń. Może również testować różne rzeczy, a wiedza opisana w grze na temat danych obiektów skoryguje jego kolejne zachowania odnośnie tych obiektów, np. jego pragnieniem może być zaspokojenie głodu, więc spróbuje zjeść napotkany kamień, jednak z opisu w grze wynika, że nie nadaje się on do jedzenia. Tym samym zaktualizuje swoje drzewo decyzyjne dotyczące głodu i już nie będzie próbował jeść kamienia.

Kolejnym sposobem nauki jest nagradzanie lub karzenie agenta, przez gracza za wykonane czynności. Oto przykład drzewa decyzyjnego jakie zostanie zbudowane po ataku na kilka wiosek oraz odpowiedniego odniesienia się do agenta. Do utworzenia drzewa wykorzystano algorytm ID3<sup>65</sup>.

Tabela 3. Numeryczna reprezentacja reakcji gracza na zachowanie agenta. [35]

Co zaatakował	Opinia gracza
Przyjazna wioska, słaba obrona, plemię Celtów	-1.0
Wroga wioska, słaba obrona, plemię Celtów	+0.4
Przyjazna wioska, mocna obrona, plemię Nordyków	-1.0
Wroga wioska, mocna obrona, plemię Nordyków	-0.2
Przyjazna wioska, średnia obrona, plemię Greków	-1.0
Wroga wioska, średnia obrona, plemię Greków	+0.2
Wroga wioska, mocna obrona, plemię Greków	-0.4
Wroga wioska, średnia obrona, plemię Azteków	0.0
Przyjazna wioska, słaba obrona, plemię Azteków	-1.0

<sup>64</sup> Ilustracja wykonana na podstawie obrazka zamieszczonego w opracowaniu.[35]

<sup>65</sup> Algorytm służący do budowy drzewa decyzyjnego, obecnie mający znaczenie historyczne, zastąpiony przez algorytm C4.5 [36].

## 4. Nowoczesna grafika komputerowa

„Grafika trójwymiarowa w ogólności jest używana do tworzenia realistycznie wyglądającego środowiska, widocznego na dwuwymiarowym ekranie.” [38]

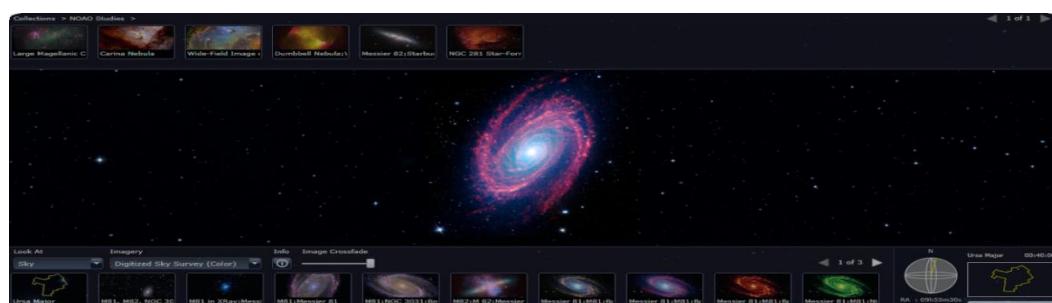
Korporacja NVIDIA<sup>66</sup>

Czemu akurat grafika 3D? Rozwój grafiki komputerowej jest odpowiedzią na coraz większe wymogi klientów. Ostatnie dziesięciolecie było świadkiem dużego kroku naprzód w zakresie wizualizacji, wydajności i interaktywności. Współczesne procesory graficzne są w stanie sprostać coraz to bardziej wymagającym i skomplikowanym techniką wizualizacji. Zwiększa to automatycznie możliwości i zakres wykorzystania grafiki trójwymiarowej (Tabela 4).

Tabela 4.Zastosowanie grafiki komputerowej<sup>67</sup>

Wizualizacja danych	Biznes
Raporty i wykres 3D	Procesy przemysłowe
Nauka i astronomia	Nieruchomości i wirtualnej wycieczki
Edukacja i szkolenie	Obsługa klienta
Marketing i reklama	Branża filmowa
Medycyna	Gry i symulacje

Wizualizacja wykracza poza obszar rozrywki znajdująca się coraz szersze zastosowanie w nauce i edukacji. Zwiększa jakość obrazowania przebiegu procesów, prezentacji danych, promocji produktów oraz sprawnej i efektywnej obsługi klienta. Przykład takiego zastosowania obrazuje Ilustracja 4.1.



Ilustracja 4.1. Wizualizacja danych: nauka i astronomia<sup>68</sup>

<sup>66</sup>Korporacja NVIDIA jest światowym liderem w dziedzinie produkcji procesorów graficznych i procesorów obsługi mediów cyfrowych.

<sup>67</sup> Źródło własne.

## 4.1 Modele i geometria

Obiekty oraz modele w grafice trójwymiarowej reprezentowane są poprzez siatki trójkątów (patrz Ilustracja 4.2). Trójkąty takie posiadają różne odcienie kolorów jak i tekstur, oraz mogą być oświetlone przez lokalne bądź globalne źródła światła.



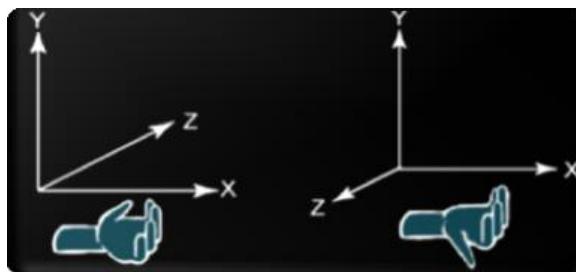
Ilustracja 4.2. Obiekty geometryczne w grafice trójwymiarowej<sup>69</sup>

Współrzędne punktów reprezentowane są jako trzy składowe wektora ( $x, y, z$ ) zwane wierzchołkami. Wierzchołki są połączone tworząc figury proste, np. takie jak trójkąty. Paski trójkątów formują siatki geometrii, z których zbudowane są obiekty (Ilustracja 4.3).



Ilustracja 4.3. Wierzchołki i ich podstawowe konstrukcje<sup>70</sup>

W większości silników graficznych wstępuje prawo skrętny układ współrzędnych, z osią  $Z$  skierowaną w obserwatora. Dwa rodzaje układów współrzędnych ukazuje Ilustracja 4.4.



Ilustracja 4.4. Lewo i prawo skrętny układ współrzędnych<sup>71</sup>

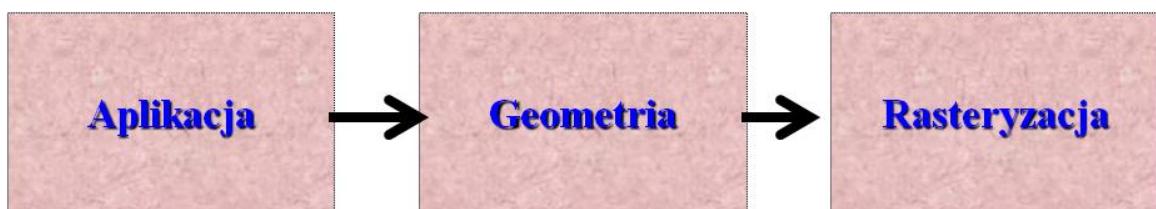
<sup>68</sup> Zdjęcie opracowane na podstawie programu C. WOng, „Interactive Tours” [37]

<sup>69</sup> Ilustracja opracowana na podstawie A. Oneal, „Silverlight 3D Graphics”[40]

<sup>70</sup> Ilustracja opracowana na podstawie A. Oneal, „Silverlight 3D Graphics”[ 40]

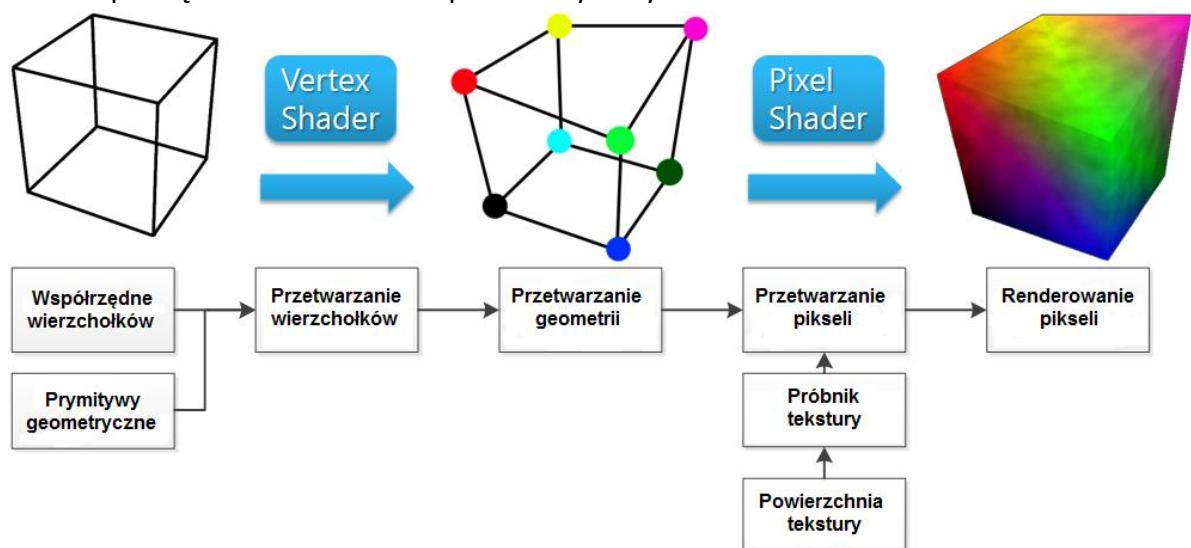
## 4.2 Interaktywne przetwarzanie obrazu w czasie rzeczywistym

Proces analizy modelu sceny trójwymiarowej oraz tworzenia na jego podstawie wyjściowego obrazu dwumiarowego nazywany jest renderowaniem (ang. rendering). Architektura potoku renderowania (ang. rendering pipeline), która przedstawia Ilustracja 4.5, składa się na trzy etapy:



Ilustracja 4.5. Architektura potoku<sup>72</sup>

Każdy etap może być wykonywany równolegle wykorzystując wielordzeniowość, przydzielone procesory, a całkowita przepustowość potoku jest równa jego najwolniejszemu etapowi. Szczegółowy schemat architektury potoku prezentuje Ilustracja 4.6. Etap „Aplikacja” zaimplementowany jest w oprogramowaniu i odbywa się w całości na procesorze komputera. Odpowiada za sczytanie danych z urządzeń wejściowych, detekcję kolizji, fizykę wizualizacji, optymalizację oraz algorytmy przyspieszające działanie. Ostatnią czynnością etapu aplikacji jest przesłanie geometrii obiektów, które mają zostać wypisane. Są to punkty, linie, trójkąty, prostokąty oraz współrzędne wierzchołków przechowywanych w buforze wierzchołków. Dane w

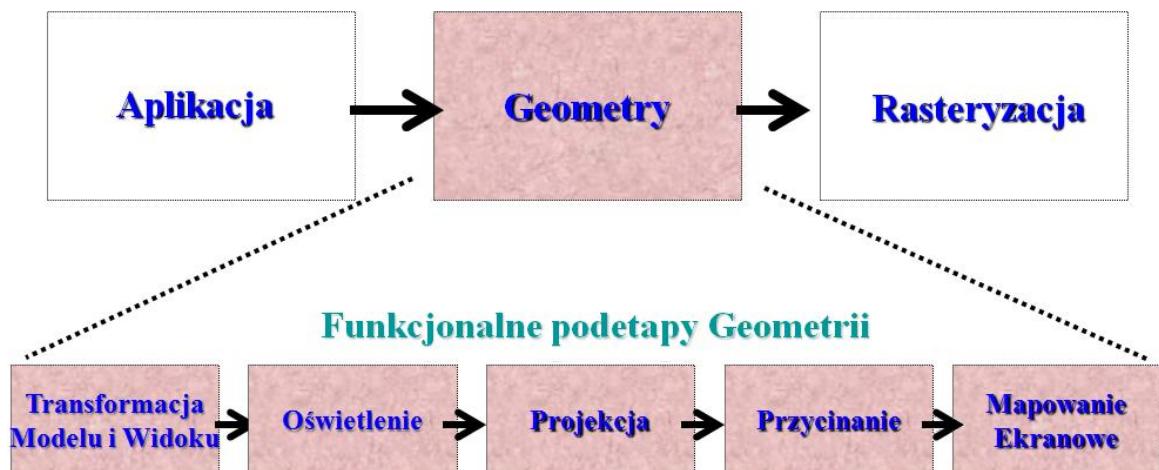


Ilustracja 4.6. Architektura potoku – Vertex Shader oraz Pixel Shader<sup>73</sup>

<sup>71</sup> Ilustracja opracowana na podstawie A. Oneal, „Silverlight 3D Graphics” [40]

<sup>72</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

buforze mogą oprócz pozycji zawierać odpowiednio dla każdego wierzchołka kolor oraz współrzędne tekstury. Bufor ten przesyłany jest do pamięci karty graficznej. Cieniowanie wierzchołkowe (ang. Vertex Shader) wykonuje transformacje wierzchołków znajdujących się w tym buforze. Proces ten jest dość wymagający obliczeniowo, dlatego w całości za jego wykonanie odpowiedzialny jest procesor karty graficznej, odciążający centralny procesor komputera. Cieniowanie (ang. Shader) jest krótkim programem napisanym w języku cieniowania (ang. Shader language). Obecnie istnieje wiele języków cieniowania, których składnia jest podobna, a w niektórych przypadkach prawie jednakowa, gdyż były opracowywane wspólnie przez różne korporacje. Przykładem takiego języka jest HLSL (ang. High Level Shading Language) – język cieniowania wysokiego poziomu biblioteki DirectX<sup>74</sup> firmy Microsoft, oraz język cieniowania Cg (ang. C for graphics – C dla grafiki) opracowany przez firmę NVIDIA w oparciu o języki programowania C i C++. Ilustracja 4.7 przedstawia etap „Geometria”, który następuje po cieniowaniu wierzchołkowym.



Ilustracja 4.7. Architektura potoku – Podetapy Geometrii<sup>75</sup>

- Transformacja modelu i widoku

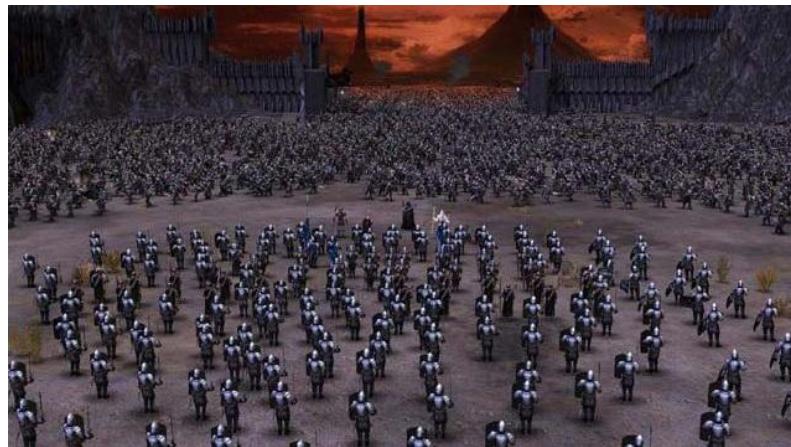
Każdy obiekt został stworzony we własnej przestrzeni modelu (ang. Model Space). Wszystkie modele podlegają transformacji rotacji, translacji lub skalowania. Umożliwia to użycie tej samej geometrii, przy odmiennych przekształceniach, do jej reprezentacji w nowej, unikatowej przestrzeni w scenie. Technika ta nosi nazwę „Geometria Chwilowa” (ang. Geometry Instancing). Dzięki niej możliwe jest zgrupowanie znacznej ilość jednakowych obiektów, w tym samym momencie. Oddziaływa to znacząco na poprawę wydajność, w szczególności gdy na scenie ma pojawić się spora liczba

<sup>73</sup> Ilustracja opracowana na podstawie A. Oneal, „Silverlight 3D Graphics” [40]

<sup>74</sup> DirectX – biblioteka firmy Microsoft wspomagająca generowanie grafiki dwu- i trójwymiarowej oraz dźwięku

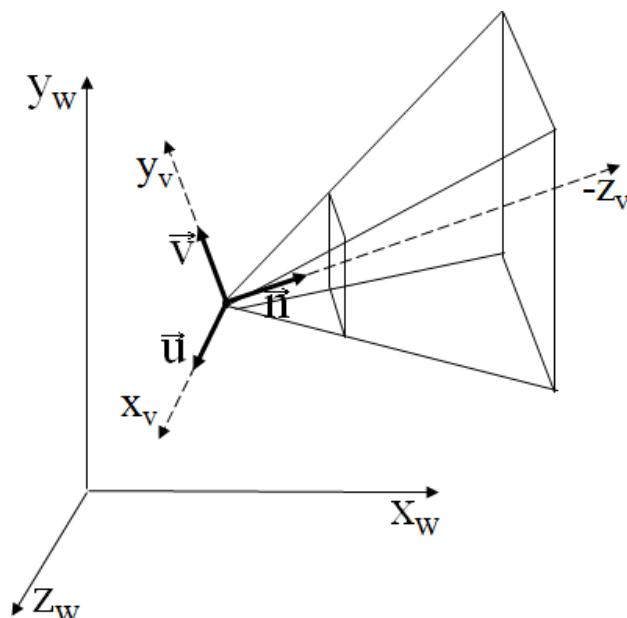
<sup>75</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

identycznych elementów takich jak drzewa, trawa, żołnierze (patrz Ilustracja 4.8), czy pojazdy mechaniczne. Modele po transformacjach znajdują się w unikalnej przestrzeni świata (ang. World Space).



Ilustracja 4.8. Przykład zastosowania geometrii chwilowej – skupisko żołnierzy<sup>76</sup>

Transformacja widoku przekształca obiekt z przestrzeni świat we współrzędne kamery, określając jej położenie. Kamera istnieje w przestrzeni wirtualnego świata, posiada swoją pozycję oraz kierunek. Jedynie obiekty przez nią widziane są wyświetlane na ekranie.



Ilustracja 4.9. Kamera w przestrzeni świata<sup>77</sup>

Ilustracja 4.9 przedstawia oko kamery, znajdujące się w centrum układu współrzędnych kamery. Patrzy w negatywnym kierunku osi  $Z_v$ , reprezentowanym przez wektor

<sup>76</sup> Ilustracja opracowana na podstawie K. Myzia „GF8800GTX, 8800GTS - rewolucja czy postęp?” [41]

<sup>77</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

normalny  $\vec{n}$ . Posiada skierowany do góry, w kierunku osi  $Y_v$ , wektor  $\vec{v}$  (ang. up vector), oraz patrzący w prawo, wzduż osi  $X_v$  wektor  $\vec{u}$  (ang. right vector).

Transformacja widoku otrzymywana jest jako iloczyn macierzy obrotu i macierzy przemieszczenia:

$$M_{vis} = R_{rot} \bullet T_{trans} \quad (65)$$

gdzie:

$$\text{macierz obrotu: } R_{rot} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ -n_x & -n_y & -n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (66)$$

$$\text{macierz przemieszczenia: } T_{trans} = \begin{bmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (67)$$

$\overset{\rightarrow}{VRP}$  - wektor przemieszczenia,

Jako wynik otrzymujemy macierz widoku  $M_{vis}$ :

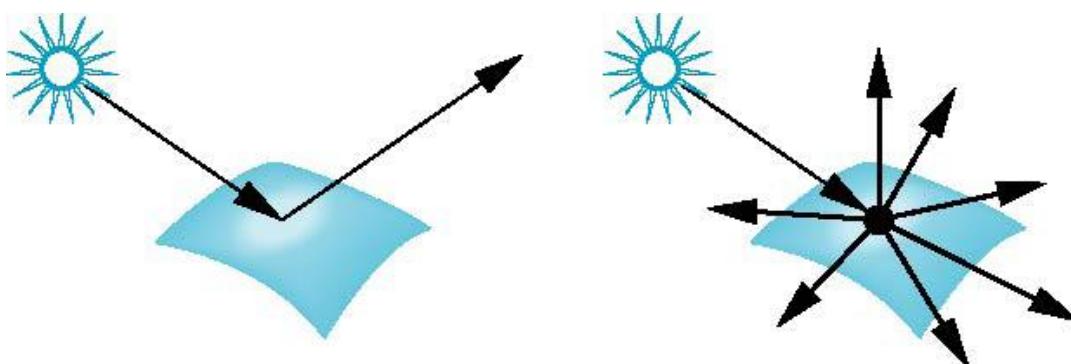
$$M_{vis} = \begin{bmatrix} u_x & u_y & u_z & -u \bullet VRP \\ v_x & v_y & v_z & -v \bullet VRP \\ -n_x & -n_y & -n_z & n \bullet VRP \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (68)$$

Na tym etapie do zwiększenia wydajności gry dla obiektów zbudowanych z wielokątów, w których określony jest „przód” wielokąta, wyznaczony przez wektor normalny jego powierzchni, nie wyświetla się powierzchni skierowanych „tyłem” do kamery. Metoda ta nosi nazwę „odrzucania tylnich ścian” (ang. backface culling), pozwala ona odrzucić znaczną część wielokątów niewidocznych dla obserwatora. Otrzymywane są one na podstawie iloczynu skalarnego normalnej powierzchnia, a wektorem widzenia kamery. Metoda ta nie gwarantuje, że wszystkie niewidoczne wielokąty zostaną odrzucone, sprawdza się ona w stu procentach tylko dla brył wypukłych. [39]

- Oświetlenie

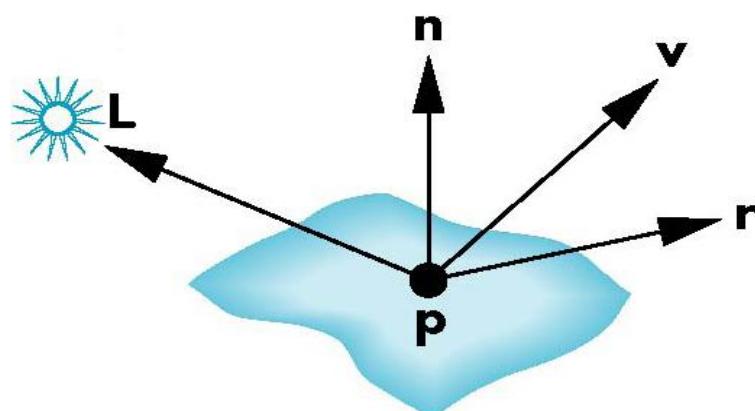
Prawidłowe przedstawienie modeli wymaga obliczenia oświetlenia jakie na nie pada. Promień światła padający na materiał modelu powoduje, że każdy jego punkt może mieć inną barwę lub odcień. Rezultatem cieniowania obiektów jest otrzymanie dużo bardziej realistycznych scen. Kolor wierzchołka obliczany jest na podstawie rodzaju

i liczby symulowanych źródeł światła, modelu oświetlenia, właściwości materiału powierzchni (patrz Ilustracja 4.10), efektów atmosferycznych takich jak mgła, czy dym. Światło padające na obiekt może zostać po części odbite, zabsorbowane lub rozproszone. Poprawne cieniowanie wymaga globalnych kalkulacji z udziałem wszystkich obiektów i źródeł światła. Jest to jednak niemożliwe w potokowej architekturze renderowania grafiki, gdzie cieniowanie każdego wielokąta odbywa się lokalnie. Istnieje jednak wiele technik uzyskujących efekt zbliżony do efektu globalnego. Rozróżniamy trzy podstawowe źródła światła: punktowe – opisywane przez pozycję i kolor; reflektor, będący ograniczonym światłem z idealnego źródła punktowego oraz światło otoczenia, charakteryzujące się identyczną ilością światła dla całej sceny.



Ilustracja 4.10. Rodzaje powierzchni – z prawej gładka, z lewej chropowata<sup>78</sup>

Im gładszym jest powierzchnia, tym bardziej odzwierciedlane światło jest skupione w kierunku doskonałego lustrzanego odbicia. Chropowate powierzchnie rozpraszają światło we wszystkich kierunkach. Ilustracja 4.11 prezentuje powszechnie znany i stosowany model oświetlenia w grafice czasu rzeczywistego, zwanym „oświetleniem Phonga”<sup>79</sup>.



Ilustracja 4.11. Model oświetlenia Phonga<sup>80</sup>

<sup>78</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

<sup>79</sup> Oświetlenie Phonga – model oświetlenia w grafice komputerowej opracowany przez Phong Bui-Tuonga w roku 1975

<sup>80</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

Model ten jest prosty i szybki do obliczenia. Składa się on na cztery wektory:

- $\mathbf{L}$  – źródło światła,
- $\mathbf{n}$  – wektor normalny powierzchni w punkcie  $\mathbf{p}$ ,
- $\mathbf{v}$  – pozycja obserwatora,
- $\mathbf{r}$  – idealne odbicie.

Natężenie światła dochodzącego do obserwatora dzieli się na:

- Natężenie światła rozproszonego (ang. diffuse) obliczane jest na podstawie modelu Lamberta – „*Ilość światła odbitego jest proporcjonalne do składowej pionowej światła przychodzącego*” [39].

$$I_d = I_i \cdot \cos \beta \quad (69)$$

gdzie:

$I_i$  - natężenie światła padającego,

$\beta$  - kąt pomiędzy światłem padającym, a normalną do powierzchni.

- Natężenie światła odbitego zwierciadlanie (ang. specular) wynosi:

$$I_s = I_r \cdot \cos \alpha \quad (70)$$

gdzie:

$I_r$  - natężenie światła odbitego,

$\alpha$  - kąt pomiędzy światłem odbitym, a pozycją obserwatora.

- Natężenie światła otoczenia (ang. ambient) jest wynikiem wielu interakcji pomiędzy źródłami światła i obiekty we środowisku. Jest ono stałe i wynosi  $I_a$ .

Każde z natężeń przemnożone jest przez odpowiadające im współczynniki absorpcji  $k_i$ , posiadające wartości z przedziału  $[0, 1]$ . Określają one procentowy wpływ składowych na wynikowe natężenie. Oświetlenie  $I_p$  w podstawowym modelu Phonga dla punktu  $p$ , wyrażone jest jako suma natężeń światła rozproszonego, odbitego zwierciadlanie oraz światła pochodzącego z otoczenia:

$$I_p = k_d I_d + k_s I_s + k_a I_a \quad (71)$$

Wzór ten może być rozwinięty o składnik emisyjny materiału, symulujący źródło światła, oraz o współczynnik tłumienia światła charakteryzujący spadek jego natężenia wraz z odległością.<sup>81</sup>

---

<sup>81</sup> Opracowane na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

$$f = \frac{1}{k_c + k_l d + k_q d^2} \quad (72)$$

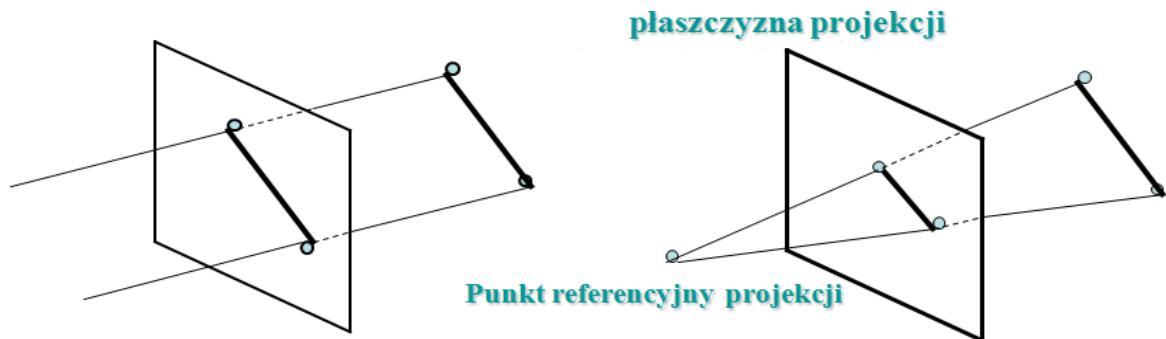
gdzie:

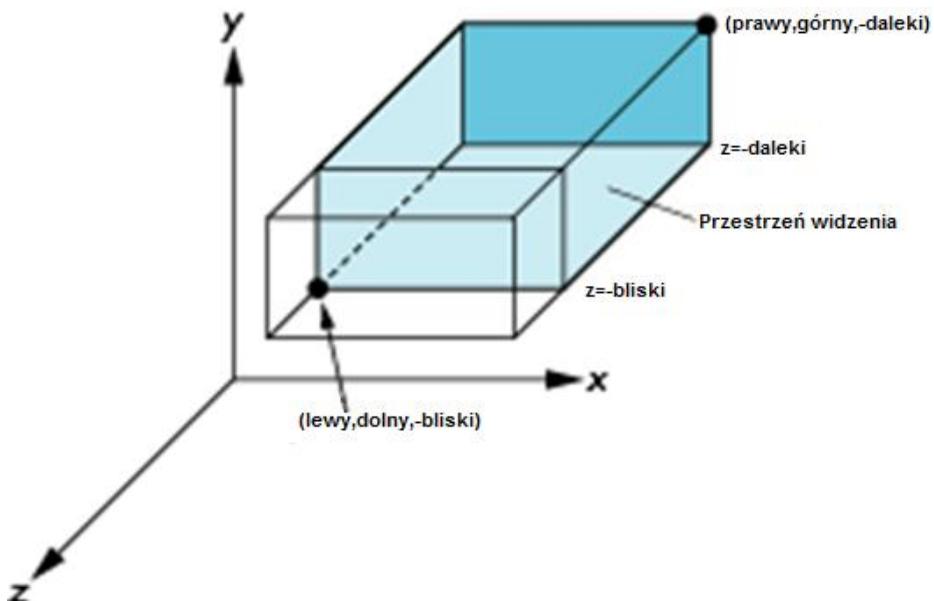
- $f$  - współczynnik tłumienia światła,
- $d$  - zasięg, w którym światło oddziałuje na obiekt,
- $k_c$  - stała tłumienia,
- $k_l$  - współczynnik tłumienia liniowego,
- $k_q$  - współczynnik tłumienia kwadratowego.

Istotnym elementem na tym etapie jest optymalizacja procesu wyliczania oświetlenia. Mniejsza ilość światła jak i prostszy model cieniowania wymaga mniejszego nakładu obliczeniowego, odbija się to jednak na realizmie przetwarzanej sceny. Wyróżnić tu należy model „*płaskiego cieniowania*” (ang. flat shading), gdzie jasność wielokąta jest jednolita i obliczana na podstawie kąta pomiędzy jego powierzchnią, a kierunkiem padającego światła. Wadą stosowania tego podejścia są zauważalne kanty i płaskie powierzchnie bryły. Główną zaletą tego modelu jest prędkość w jego przetwarzaniu, gdyż na każdą ścianę bryły wykonywane jest jedno obliczenie. W modelu Phonga, kalkulacje wykonywane są dla każdego piksela. Inny model oświetlenia, znany pod nazwą „*cieniowania Gourauda*” (ang. Gouraud shading) oblicza oświetlenie dla każdego wierzchołka bryły, a jasność poszczególnego piksela wielokąta otrzymywana jest poprzez interpolację. [39]

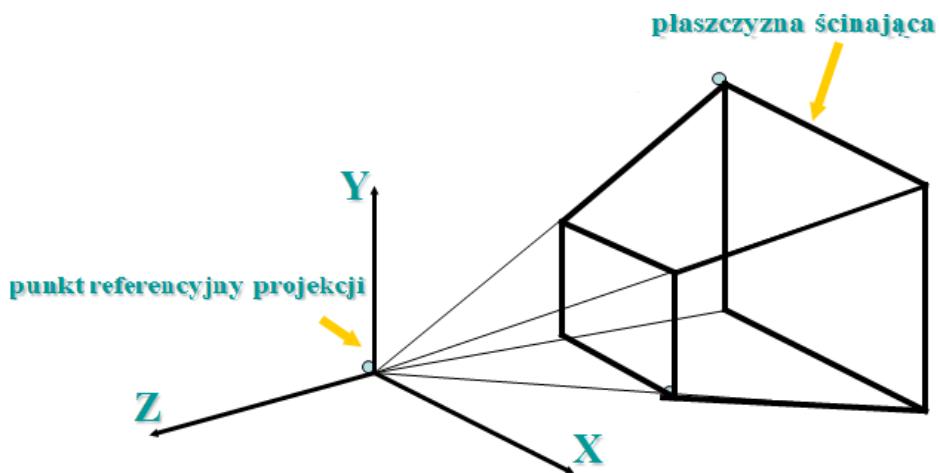
- Projekcja

Etap ten modeluje, jaką objętość przestrzeni wirtualnego świata jest widziany przez kamerę. Ilustracja 4.12 przedstawia dwa rodzaje projekcji: rzutowanie równoległe, zwane ortograficznym, gdzie linie równoległe, pozostają równoległe; oraz rzutowanie perspektywiczne, uwzględniające perspektywę.



Ilustracja 4.12. Rodzaje projekcji, z lewej równoległa, z prawej perspektywiczna<sup>82</sup>Ilustracja 4.13. Przestrzeń ortogonalna<sup>83</sup>

Każdy punkt przestrzeni ortogonalnej, który obrazuje Ilustracja 4.13, posiada swoje odwzorowanie na wybranej płaszczyźnie, poprzez prostą prostopadłą do tej płaszczyzny, przechodzącą przez dany punkt. Wyłącznie obiekty znajdujące się wewnątrz przestrzeni widzenia, pomiędzy jej płaszczyznami, są widoczne przez kamerę. Projekcja perspektywicznej przestrzeń widzenia, którą przedstawia Ilustracja 4.14, nosi nazwę „ostrosłupa ścieżego” (ang. Frustum). Cześć wirtualnego świata widzianego przez kamerę ścieżą jest poprzez płaszczyznę przednią i tylną.

Ilustracja 4.14. Ostrosłup widzenia w przestrzeni perspektywicznej<sup>84</sup>

<sup>82</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

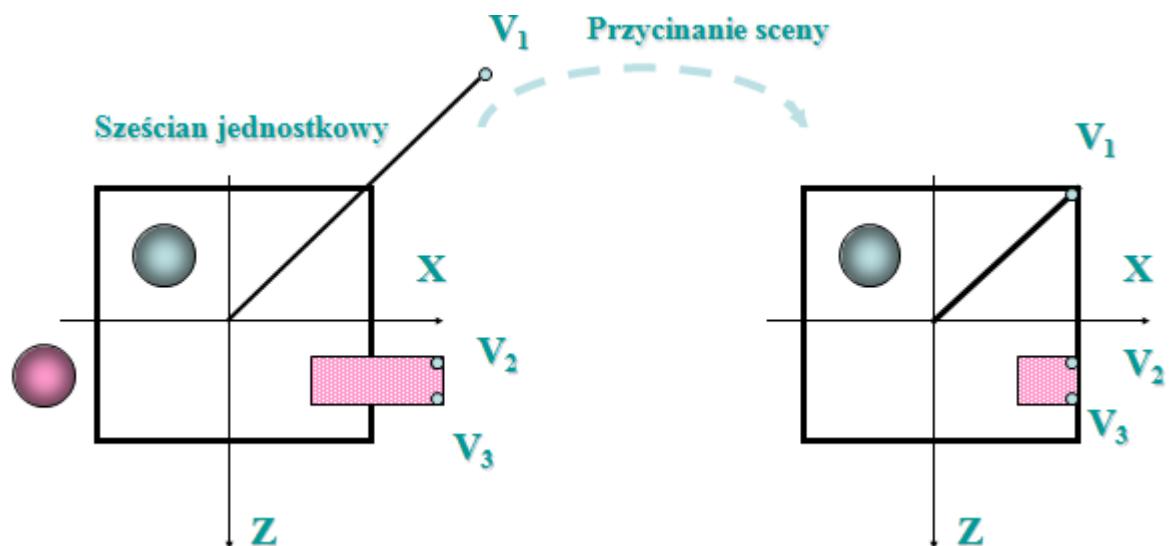
<sup>83</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

<sup>84</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

Przestrzeń taka ulega kanonicznemu mapowaniu, jest to perspektywistyczne przekształceniu obszaru widzenia do sześcianu jednostkowego. Innymi słowy, jest to kanoniczne znormalizowanie przestrzeni widzenia. [39]

- Przycinanie

Jedynie obiekty znajdujące się całkowicie lub częściowo w ostrosłupie widzenia będą przesłane do ostatniego etapu potoku przetwarzania - rasteryzacji. Są one mapowane do sześcianu jednostkowego, a następnie poddawane procesowi „przycinania” (ang. clipping). Geometria poza przestrzenią przecięcia jest odrzucana, a obiekty leżące na granicy zostają przycięte. Ilustracja 4.15 przedstawia działanie tego procesu, który jest realizowany przez dodanie nowych wierzchołków, składających się na nową krawędź w miejscu cięcia.



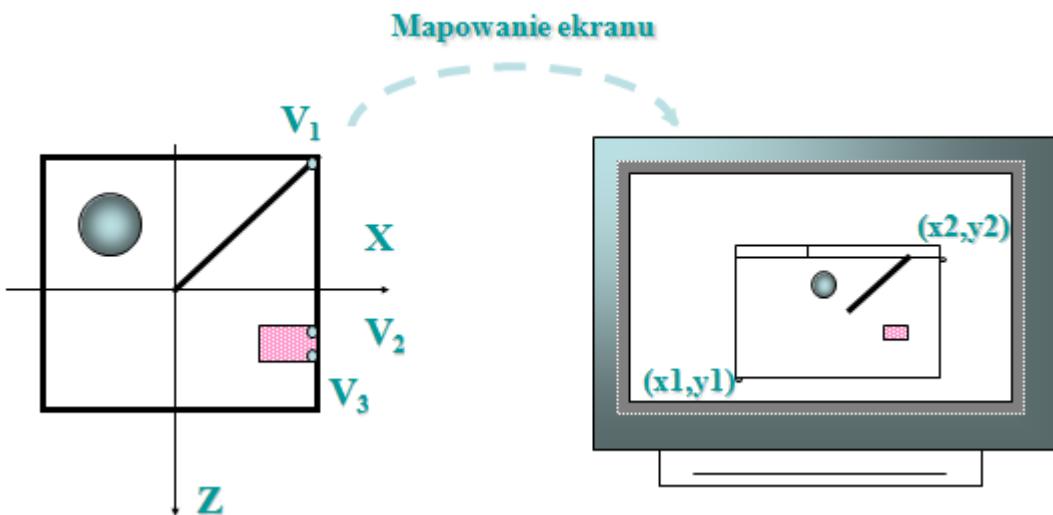
Ilustracja 4.15. Transformacja przycinania sceny<sup>85</sup>

Na powyższym przykładzie wierzchołek  $V_1$  jest zamieniony przez nowy, w miejscu przecięcia się linii i płaszczyzny przecięcia. Identyczna sytuacja zachodzi dla wierzchołków  $V_2$  oraz  $V_3$ . [39]

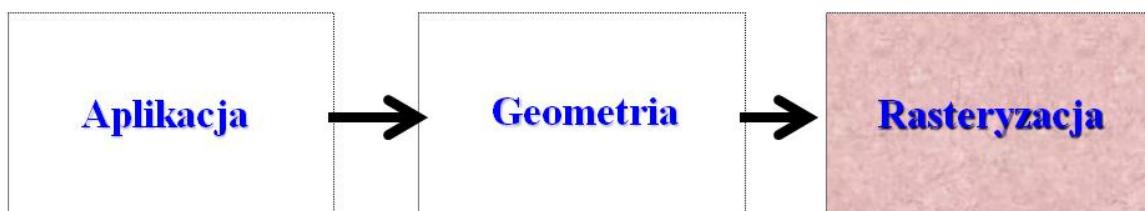
- Mapowanie ekranu

Przycięte obiekty geometryczne transformowane są do przestrzeni okna. Proces ten polega na skalowaniu oraz transpozycji, wpływających na współrzędne  $x$  i  $y$  obiektów, lecz nie na współrzędną  $z$ . Tak przebrane obiekty we współrzędnych ekranowych przesyłane są do końcowego etapu potoku renderowania. Ilustracja 4.16 obrazuje przebieg mapowania ekranu.

<sup>85</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

Ilustracja 4.16. Mapowanie ekranu<sup>86</sup>

Rasteryzacja jest ostatnim etapem potoku renderowania (Ilustracja 4.17). Jej celem jest obliczenie poprzez interpolację kolorów pikseli pokrywających przesłane z etapu geometrii obiekty. Dwuwymiarowe wierzchołki w przestrzeni ekranu, wraz z ich wartościami wymiaru  $Z$ , konwertowane są do przestrzeni pikseli na ekranie monitora.

Ilustracja 4.17. Architektura potoku – rasteryzacja<sup>87</sup>

Dla każdej próbki powierzchni obiektu tworzony jest fragment. Wartości parametrów fragmentu obliczane są poprzez interpolację wartości otaczających go wierzchołków. Informacje o kolorze każdego piksela przechowywane są w buforze kolorów. Kolejnym krokiem etapu rasteryzacji jest łączenie wygenerowanych fragmentów z odpowiadającymi im wartościami w buforze koloru. W tym celu przeprowadzane są kolejno testy:

- Alfa test

Przenikanie alfa (ang. alpha blending) umożliwia wyświetlanie półprzeźroczystych obiektów. O stopniu przeźroczystości każdego piksela decyduje jego wartość alfa, która jest przechowywana w kanale alfa (ang. alpha channel). W przypadku występowania tła nieprzeźroczystego o wartościach kanałów  $(R_0, B_0, G_0, 1)$ , oraz znajdującego się przed

<sup>86</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

<sup>87</sup> Ilustracja opracowana na podstawie J. Madeiras Pereira, „The 3D Graphics Rendering Pipeline” [39]

nim fragmentu z wartościami  $(R_1, B_1, G_1, \alpha_1)$  wartość koloru wynikowego  $(R'_1, B'_1, G'_1)$  obliczana jest następująco:

$$\begin{aligned} R'_1 &= \alpha_1 R_1 + (1 - \alpha_1) R_0 \\ G'_1 &= \alpha_1 G_1 + (1 - \alpha_1) G_0 \\ B'_1 &= \alpha_1 B_1 + (1 - \alpha_1) B_0 \end{aligned} \quad (73)$$

Alfa test wykonywany jest przed testem widoczności. Porównywana jest wartość alfa z wcześniej predefiniowaną, stałą wartością. Umożliwia to automatyczne odrzucenie całych fragmentów, które nie przeszły testu. Niezapisywane są półprzeźroczyste piksele do bufora Z, w wyniku czego otrzymujemy lepszą wydajność. [39]

- Test szablonu<sup>88</sup>

Test ten porównuje wartości odpowiednich fragmentów z wartościami z dodatkowego bufora (ang. stencil buffer). Stosując odpowiednie operatory do bufora, możliwe jest odrzucenie fragmentów obrazu z dalszego przetwarzania w potoku renderowania.

- Test widoczności<sup>89</sup>

Wartości osi wymiaru Z każdego piksela przechowywane są w buforze Z. Dla każdego fragmentu sprawdzane jest, czy znajduje się on bliżej ekranu, porównując jego wartość Z, z aktualnie występującą w buforze Z. Jeśli znajduje się bliżej, to nadpisywana jest istniejąca wartość bufora, w przeciwnym wypadku bufor pozostawiany jest bez zmian, gdyż fragment jest niewidoczny – zasłonięty przez inny obiekt na scenie. W przypadku występowania obiektów przeźroczystych, wymagane jest sortowanie od najdalszego do najbliższego obiektu na scenie (ang. back-to-front sorting). [39]

W końcowym etapie procesu rasteryzacji, wygenerowany obraz trafia do bufora tylnego (ang. back buffer). W chwili odrysowania sceny, bufor tylny zamieniany jest zawartością z buforem przednim (ang. front buffer) i przesyłany jest do wyświetlacza.

### 4.3 Odwzorowanie fizyki w nowoczesnej grafice komputerowej

Ross Nordby w swoim wykładzie o silniku fizyki [42] definiuje symulacje fizyczną jako proces przybliżający rzeczywistość. W grafice komputerowej silnik fizyki jest to narzędzie umożliwiające symulowanie zjawisk fizycznych z dużą wydajnością i wysokim stopniem realizmu. Rozszerza to możliwości interakcyjne użytkownika ze światem.

---

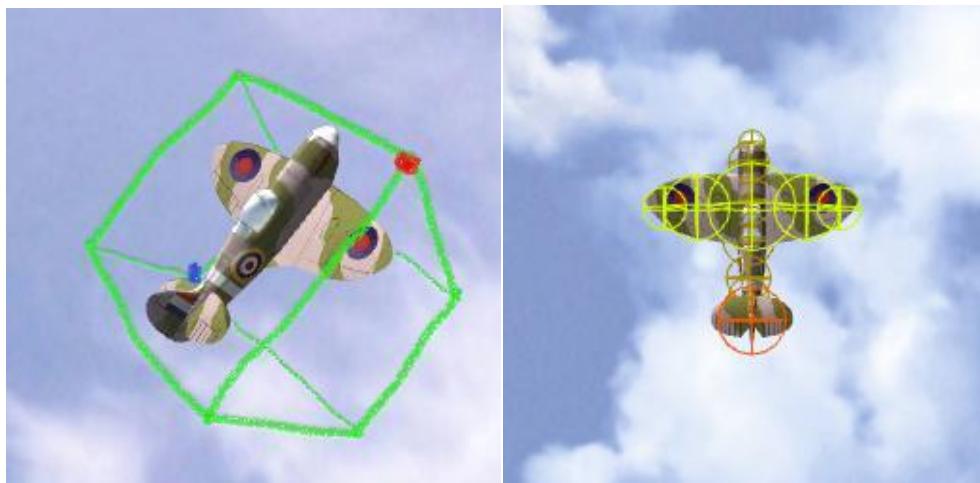
<sup>88</sup> Stencil (ang.) – szablon, matryca

<sup>89</sup> Ang. Depth-test

W dzisiejszych czasach przeciętny komputer jest w stanie obsłużyć od tuzinów po setki, symulowanych w czasie rzeczywistym, jednostek fizycznych w grafice trójwymiarowej, utrzymując wysoką wydajność obrazu.

Możliwa jest symulacja powszechnych zachowań, ruchów, wybuchów, trajektorii lotu, zderzeń, przemieszczania się w nieregularnym terenie. W tym celu silnik fizyki dostarcza interfejs nadający własności fizyczne obiektom, umożliwiający oddziaływanie grawitacyjne, ruchy prostoliniowe i przyspieszone, obsługujący detekcję kolizji, dający kontrolę czasu oraz wydajności w zależności od możliwości sprzętowych.

Powszechną metodą wykrywania zderzeń jest stosowanie brył brzegowych (ang. bounding volume), które w uproszczony sposób opisują najmniejszą przestrzeń, całkowicie zawierającą dane obiekty. Gdy bryły brzegowe dwóch obiektów pokrywają się dochodzi do kolizji, która w odpowiedni sposób może być obsłużona. Ilustracja 4.18 przedstawia dwa, powszechnie stosowane typy brył brzegowych.



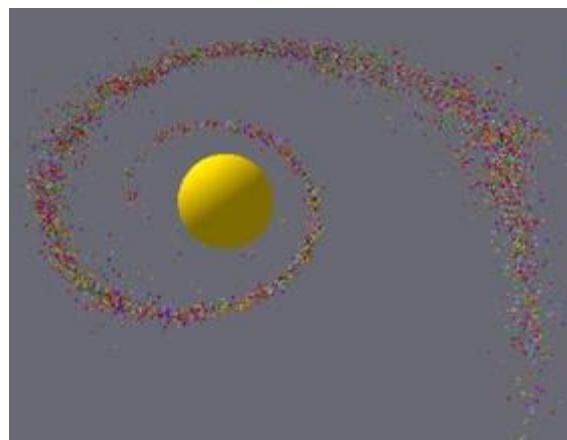
Ilustracja 4.18. Z lewej pudełko brzegowe<sup>90</sup>, z prawej kula brzegowa<sup>91</sup> [43]

Poruszając tematykę symulacji fizycznych należy wspomnieć o wydajności oraz związanej z nią optymalizacją. Obliczenia i kalkulacje, w szczególności te bardziej skomplikowane, z udziałem kilkuset – kilku tysięcy ruchomych obiektów, mogą mocno obciążyć procesor komputera i znacznie zwolnić jego pracę. Ilustracja 4.19 obrazuje przykład takiej sytuacji.

---

<sup>90</sup> Ang. Bounding box

<sup>91</sup> Ang. Bounding sphere



Ilustracja 4.19. Osiem tysięcy orbitujących sześciianów<sup>92</sup>

Z pomocą przychodzą różne techniki i rozwiązania zwiększające wydajność komputera i polepszające końcową jakość symulacji. Przykładem może być zastosowanie mechanizmu wielowątkowości. Podział pracy poszczególnych komponentów na wiele części i ich rozmieszczenie na różne wątki może przyspieszyć działanie wizualizacji. Jednak i tutaj należy pamiętać o problemach komunikacji pomiędzy wątkami, synchronizacja danych, unikania zastojów. Optymalizacja prowadzi do złożoności, uciążliwsza staje się kontrola błędów, dlatego powinna być stosowana tylko wtedy kiedy jest wymagana, gdyż celem jest „wystarczająco dobry” rezultat. Wraz z postępem technologii symulacje fizyczne, w trójwymiarowej grafice komputerowej czasu rzeczywistego, będą coraz to bardziej powszechnie i praktyczne.

---

<sup>92</sup> Ilustracja opracowana na podstawie R. Nordby, „Introduction to BEPU physics and physics simulation” [42]

## 5. Koncepcja inteligentnego i autonomicznego robota

*"Inteligencja jest tą przyprawą, która z najpospolitszych rzeczy robi cuda"*

Maria Dąbrowska<sup>93</sup>

Współczesne gry komputerowe są bezwątpieni kompleksowymi systemami, których popularność nie zależy już tylko na ekskluzywnej grafice, a coraz bardziej na zaawansowanej, odwzorowującej rzeczywistość inteligencji, która dostarcza graczom więcej rozrywki, poprzez bardziej realistyczną interakcję z otaczającym ich środowiskiem gry. Gracze poszukują oderwania się od rzeczywistego świata w grach, spełnienia, a gry zaspakajają te potrzeby poprzez zapewnienie im odpowiednich wyzwań i emocji. Istnieją znakomite proste gry, jak np. „Tetris”<sup>94</sup>, która uzyskała ten efekt bez wyrafinowanej fabuły czy zaimplementowanej inteligencji. Jednakże w większości przypadków wymaga oczekującego na gracza wyzwania, w komputerowym świecie zamieszkałym przez wirtualne stworzenia, musi zostać spełniony. Powszechną praktyką w projektowaniu gier jest implementacja skryptów do kontroli fabuły i zapewnienia postaciom podstawowego zachowania. Konstrukcja taka działa szczególnie dobrze w sytuacjach, gdy wymagana fabuła jest dość sztywna. Odbywa się to kosztem elastyczności zachowań i całkowitą złożonością projektu. Projektowanie gry oparte na inteligentnych agentach zapewnia doskonałą elastyczność – każda istota staje się niezależną jednostką posiadającą osobiste postrzeganie otaczającego ją środowiska, potrafiącą podejmować swoje własne decyzje, wykonując przy tym odpowiednie akcje. Takie podejście umożliwia łatwą modyfikację zachowań jednostki, a nawet pozwala na jej uczenie się, w rzeczywistym czasie gry. Jako że użytkownik postrzega postacie z gry jako inteligentne istoty, tworzenie ich jako takie od samego początku, ułatwi sprostaniu temu oczekiwaniu w dalszych etapach rozwoju gry.

### 5.1 Założenia projektowe gry

Ideą gry jest inteligentna, autonomiczna jednostka, poruszająca się w wirtualnym świecie, posiadająca swoje cele i zadania do wykonania. Tytułowym "Robotem" jest

---

<sup>93</sup> Maria Dąbrowska (1889–1965) – polska powieściopisarka i publicystka.[44]

<sup>94</sup> „Tetris” – Puzzlowa gra video znana na cały świat, oryginalnie zaprojektowana i zaprogramowana przez Alexey Pajitnova

czołg, wykorzystujący zaimplementowaną inteligencję oraz system decyzyjny do przetrwania w nieprzyjacielskim środowisku. W grze wdrożone są następujące założenia projektowe:

- Fabuła gry rozgrywa się w trójwymiarowym świecie, zaprojektowanym i stworzonym z wykorzystaniem nowoczesnego silnika graficznego XNA;
- Podstawową jednostką w grze jest czołg, będący inteligentnym agentem, którego system rozumowania oparty jest na rozmytym wnioskowaniu;
- Głównym zadaniem czołgu jest przetrwanie i obrona monumentu orła przed innymi, wrogimi czołgami;
- Do wykonania części modeli użyte jest oprogramowanie SoftImage XSI;
- Za wizualne efekty specjalne – wybuchy, kurz, eksplozje, odpowiedzialny jest trójwymiarowy system cząsteczek;
- Kolizje, tor lotu pocisków, zderzenia oraz inne fizyczne aspekty gry obsługiwane są przez silnik fizyki.

## 5.2 Koncepcja nowoczesnej grafiki komputerowej w grze

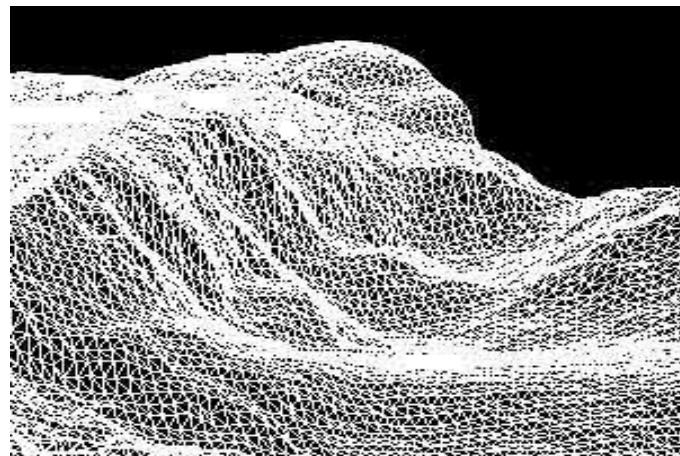
W tym dziale omówiona zostanie koncepcja wykorzystania nowoczesnej grafiki komputerowej w tworzonej grze, w oparciu o przedstawione założenia projektowe. Skoncentrujemy się na zaprojektowaniu i koncepcji budowy trójwymiarowego środowiska gry, przyjrzymy się bliżej stworzonym i wykorzystanym przez nas modelom i geometrii, ich animacji i poruszaniu się, zapoznamy się ze strukturą systemu efektów trójwymiarowych i silnikiem do obsługi oddziaływań fizycznych w grze.

- Konstrukcja wirtualnego środowiska gry

Podstawą wirtualnego świata gry jest teren, po którym mogą się poruszać występujące w niej jednostki. W celu zwiększenia realizmu i poziomu trudności, projektowany obszar będzie posiadał górzysty charakter. Dane wejściowe, które posłużą do stworzenia wyboistego terenu, składają się w całości z tekstury. Taka bitmapa, będzie zawierała piksele o kolorze w skali szarości, które będą konwertowane na wysokości poszczególnych wierzchołków siatki geometrycznej.<sup>95</sup> Ilustracja 5.1 prezentuje efekt końcowy tego procesu.

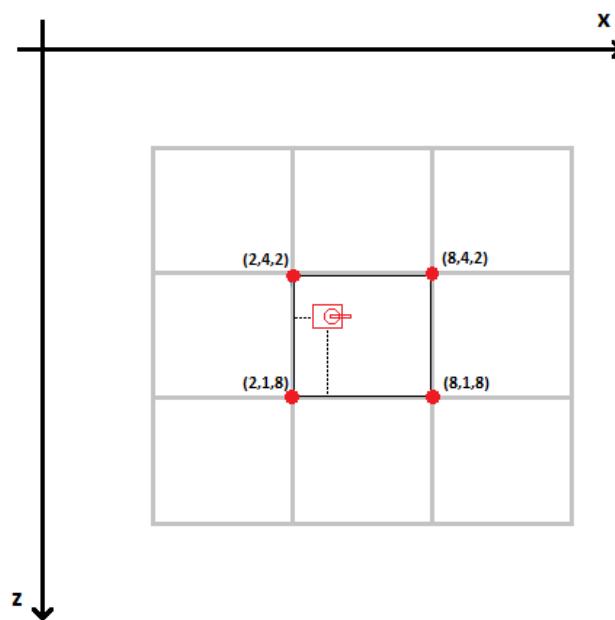
---

<sup>95</sup> 24 bitowy zapis koloru daje po 8 bitów na każdą ze składowych RGB, które przyjmują wartości z zakresu 0 – 255. Wartości te są normalizowane i mapowane na wysokość danego wierzchołka.



Ilustracja 5.1. Szkielet siatki geometrycznej tworzącej górzysty teren. [45]

Tak przygotowana geometria wymaga wyłącznie nałożenia odpowiedniej tekstury. Kolejnym krokiem jest umożliwienie obiektom poruszania się po tak stworzonym obszarze. W tym celu konieczna będzie możliwość szybkiego wyliczenia wysokości każdego punktu terenu. Ilustracja 5.2 pokazuje siatkę terenu widzianą z góry wraz ze znajdującym się na niej czołgiem.



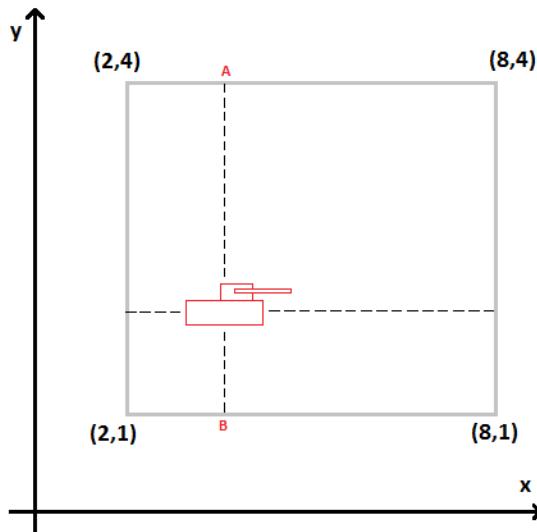
Ilustracja 5.2. Położenie czołgu – widok z góry.<sup>96</sup>

Do obliczenia wysokości czołgu zostanie użyta interpolacja biliniowa.<sup>97</sup> Znane są współrzędne  $X$ ,  $Z$  oraz wysokość  $Y$  wszystkich wierzchołków. Dostępna jest też pozycja czołgu na płaszczyźnie  $XZ$ . Bazując na przykładzie, który przedstawia Ilustracja 5.3, posługujemy się interpolacją liniową do obliczenia wartości wysokości w punkcie A.

<sup>96</sup> Ilustracja opracowana w oparciu o artykuł „Collision with a Heightmap” [46].

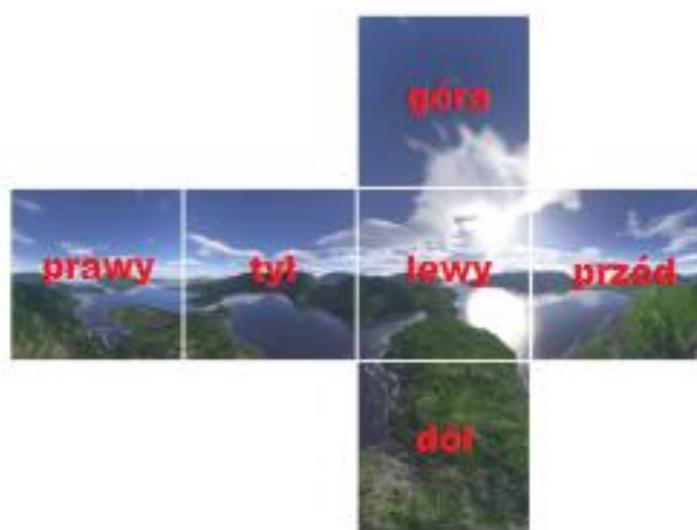
<sup>97</sup> Polega ona na obliczeniu wartości pomiędzy dyskretnymi punktami na płaszczyźnie.

Analogicznie tą samą czynność wykonamy dla punktu  $B$ .<sup>98</sup> Teraz znając wysokości w punkcie  $A$  i  $B$ , które znajdują się na prostej linii wraz z czołgiem, wykonujemy jeszcze jedną interpolację liniową pomiędzy tymi punktami otrzymując wartość wysokości czołgu.



Ilustracja 5.3. Położenie czołgu – widok z boku.<sup>99</sup>

Następnym krokiem w budowie wirtualnego świata jest wypełnienie tła. W tym celu posłużymy nam „Skybox” (pol. „pudełko nieba”), metoda uzyskiwania realistycznego tła, za pomocą dużego sześcianu, którego ściany pokryte są tekturem nieba, odległych gór lub innych krajobrazów. Ilustracja 5.4 przedstawia przykład takiej kostki.



Ilustracja 5.4. Sześciian reprezentujący tło.<sup>100</sup>

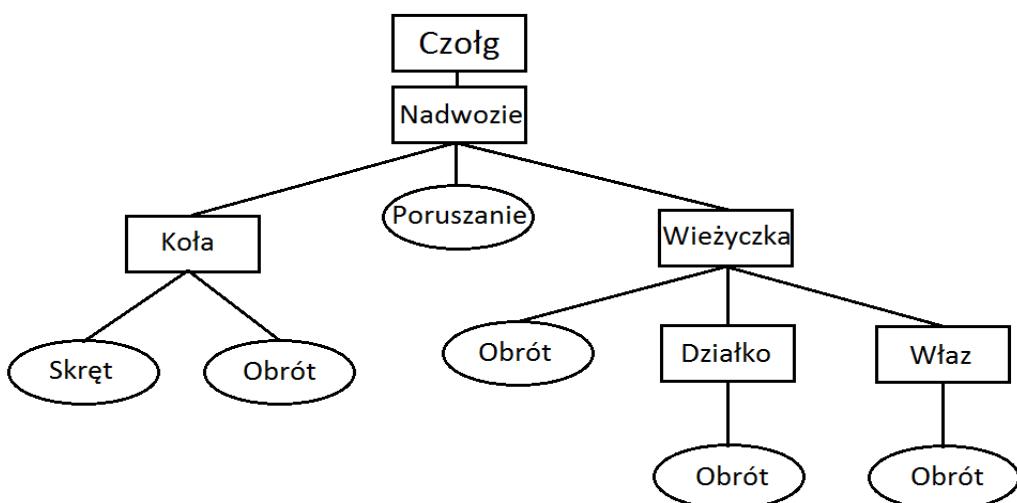
<sup>98</sup> Przykład ten jest uproszczony, gdyż wysokość Y wierzchołków krawędzi górnej jest równa. Ta sama sytuacja występuje dla wierzchołków krawędzi dolnej. W rzeczywistości wysokości wszystkich wierzchołków w większości przypadków są różne.

<sup>99</sup> Ilustracja opracowana w oparciu o tutorial „Collision with a Heightmap” [46].

Cały teren wraz z kamerą umieszczone są w środku sześcianu, i z każdym ruchem kamery, przemieszcza się on wraz z nią, dając złudzenie dużego dystansu pomiędzy obserwatorem, a odległym krajobrazem.

- Modele i geometria

Po zbudowaniu podstawowych elementów wirtualnego świata, takich jak teren i niebo, przychodzi kolej na bardziej szczegółowe obiekty. Głównym z nich jest model czołgu. Zbudowany jest on hierarchicznie z rozłącznych siatek geometrii. Koła, wieżyczka, działało oraz pozostałe elementy czołgu stanowią osobne części modelu. Każda z nich jest podłączona do odpowiedniej siatki rodzica z odpowiednim punktem obrotu. Dla przykładu, działało czołgu może podnosić się i opuszczać. Jest ono podłączone do wieżyczki, która może obracać się w prawo oraz lewo, a ona podłączona jest do nadwozia czołgu. Z powodu hierarchii, wraz z obrotem wieżyczki, obraca się również działało. Łączenie się odpowiednich części czołgu możliwe jest dzięki „kościom” (ang. bones). Schemat hierarchii zależności wraz z możliwymi transformacjami przedstawia Ilustracja 5.5.



Ilustracja 5.5. Schemat hierarchii geometrii czołgu<sup>101</sup>

W celu urozmaicenia gry oraz zwiększenia poziomu trudności w poruszaniu się po terenie, oprócz czołgów, na mapie znajdować się będą nieruchome obiekty statyczne oraz dynamiczne, mogące wchodzić w interakcje z czołgami. Do statycznych obiektów należy zaliczyć:

- budynek kościoła, umieszczony na środku mapy,
- niezniszczalne ściany, ograniczające obszar poruszania się czołgów,

<sup>100</sup> Ilustracja opracowana w oparciu o tutorial „Skybox tutorial” [47].

<sup>101</sup> Źródło własne.

- monument orła, główny cel wrogich jednostek,
- losowo rozrzucone drzewa, stanowiące główna przeszkodę w poruszaniu się czołgów.

W skład dynamicznych modeli oprócz czołgów wchodzą:

- ściany, ulegające zniszczeniu pod wpływem kolizji lub eksplozji pocisku,
- pociski wystrzelane przez czołgi,
- losowo spadające na teren apteczki zwiększające ilość punktów życiowych czołgów.

Oprócz zwykłych modeli, gra wyposażona jest w szereg geometrii wspomagającej, punkty, linie, prostopadłościany, kule, prostopadłościany, czy koła. Służą one wizualizacji zasięgu obszarów, rozmiaru przestrzeni, czy kierunku poruszania. Umożliwia to zobrazowanie działania sztucznej inteligencji w grze.

- Niekonwencjonalna koncepcja kamery w grze

Wprowadzona w grze kamera perspektywistyczna ma charakter pościgowy, połączona jest abstrakcyjną sprężyną z obserwowanym przez nią obiektem. Sprzęyna ciągnie kamerę w kierunku pożądanej pozycji. Dokładne zachowanie sprężyny zależy zarówno od jej sztywności oraz tłumienia, jak i od masy, położenia i prędkości kamery. Siłę sprężyny obliczamy korzystając z prawa Hooke'a<sup>102</sup>:

$$F_H = -k * x \quad (74)$$

gdzie:

- $k$  - jest współczynnikiem sprężystości, określającym sztywność sprężyny,  
 $x$  - jest określany jako zmiana długości:

$$\text{naprężenie} = \text{pozycja_kamery} - \text{pozycja_pozadana} \quad (75)$$

Stosując tylko tą siłę bez brania pod uwagę oporu powietrza, tarcia wewnętrznego, i innych sił działających na poruszający się obiekt, sprężyna oscylowałaby nieskończennie. Wzór na siłę rozszerzamy o siłę tłumienia:

$$F = -k * x - b * v \quad (76)$$

gdzie:

- $b$  - jest współczynnikiem proporcjonalności, określającym tłumienie,  
 $v$  - jest prędkością poruszania się obiektu.

---

<sup>102</sup> „Naprężenie wewnętrzne jest proporcjonalne do powodującego go odkształcenia”. [49]

Z drugiego prawa Newton'a<sup>103</sup> obliczamy przyspieszenie  $a$ , jako iloraz siły jaką działa sprężyna na kamerę do jej masy. Następnie obliczamy prędkość kamery w następnym kroku czasowym, stosując podstawową metodę Eulera<sup>104</sup>:

$$v_{t+\Delta t} = v_t + a * \Delta t \quad (77)$$

gdzie:

$v_{t+\Delta t}$  - jest to prędkość kamery w następnym kroku czasowym,

$\Delta t$  - krok czasowy,

$a$  - przyspieszenie kamery.

Ostatecznie obliczamy położenie  $s_{t+\Delta t}$  kamery w następnym kroku czasowym:

$$s_{t+\Delta t} = s_t + v_{t+\Delta t} * \Delta t \quad (78)$$

Tak skonstruowana kamera odwzorowuje bardziej realistyczne poruszanie się czołgu. Kiedy czołg przyspiesza, kamera cofa się, kiedy czołg skręca, w opóźnionym ruchu podąża za czołgiem, chwilowo ukazując jego profil boczny. Kiedy czołg zatrzyma się, kamera powoli odsuwa się na swoje miejsce.<sup>105</sup>

- Trójwymiarowy system cząsteczek

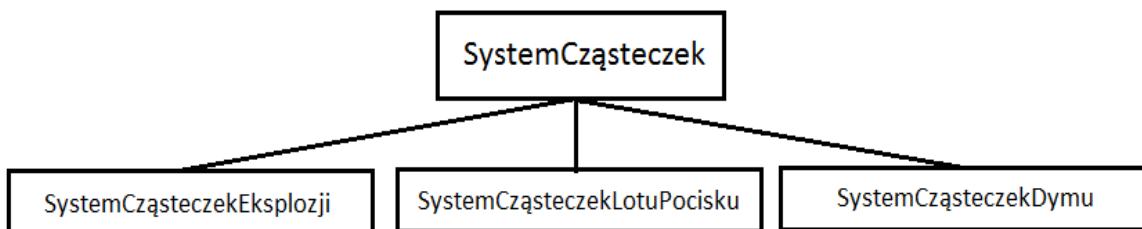
Za wizualne efekty trójwymiarowe odpowiada system cząsteczek. Animuje on cząsteczki w całości na karcie graficznej przy użyciu cieniowania wierzchołkowego. Dzięki czemu możliwe jest narysowanie ogromnej ilości cząsteczek z minimalnym obciążeniem głównego procesora komputera. Każda nowo stworzona cząsteczka ze swoją pozycją, prędkością, oraz czasem stworzenia przesyłana jest do karty graficznej. Podczas rysowania, procesor karty graficznej oblicza wiek każdej cząsteczki, porównując czas stworzenia oraz aktualny czas. Znając pozycję i prędkość początkową, oraz wiek cząsteczki, karta graficzna może obliczyć aktualną pozycję i narysować cząsteczkę w tym miejscu. Główny procesor komputera odpowiedzialny jest tylko za dodawanie nowych i usuwanie starych cząsteczek z bufora karty graficznej, ale w zupełności nie zajmuje się aktywnymi cząsteczkami. Dzięki tej technice tworzenia cząsteczek, możliwe jest rysowanie tysięcy cząsteczek, przy niskim wykorzystaniu procesora komputera.

System cząsteczek rozdzielony jest na podsystemy (Ilustracja 5.6), gdzie każdy podsystem odpowiedzialny jest za inny rodzaj symulowanych cząsteczek.

<sup>103</sup> Jeśli siły działające na ciało nie równoważą się, to ciało to porusza się z przyspieszeniem, którego wartość jest wprost proporcjonalna do siły wypadkowej, a odwrotnie proporcjonalna do masy ciała. [50]

<sup>104</sup> Metoda rozwiązywania równań różniczkowych zwyczajnych.

<sup>105</sup> Koncepcja opracowana w oparciu o tutorial „Chase Camera” [51]

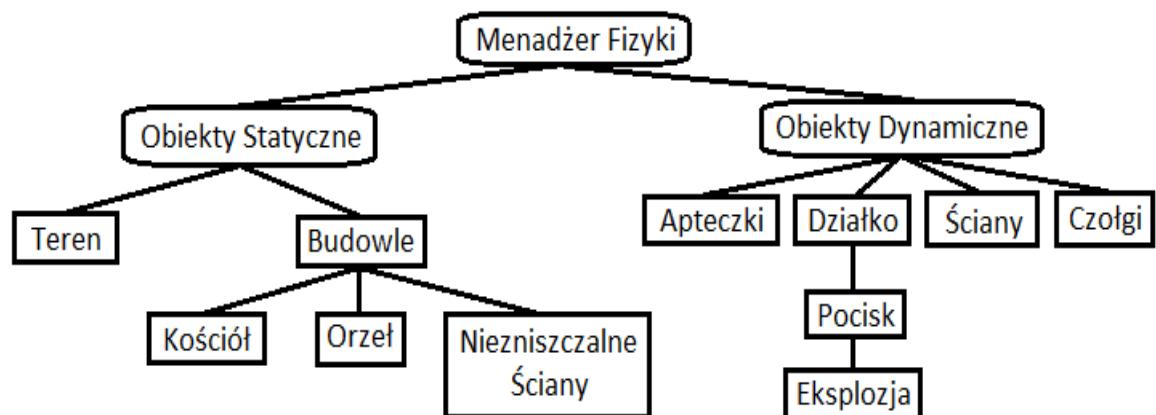
Ilustracja 5.6. Schemat systemu cząsteczek<sup>106</sup>

Dodatkowo abstrakcyjne narzędzia emisji cząsteczek, takie jak „Emiter”, czy „Pocisk” umożliwiają kombinacyjne wykorzystanie podsystemów cząsteczek, do tworzenia własnych, bardziej skomplikowanych sekwencji ich symulacji.

- Silnik fizyki

Do implementacji fizyki gry wykorzystany został silnik fizyki BEPU<sup>107</sup>. Posiada rozbudowaną bibliotekę umożliwiającą tworzenie realistycznych interakcji obiektów ze światem. Każda jednostka (ang. entity) posiada podstawowe właściwości fizyczne, jak masa, położenie, prędkość, może wchodzić w korelacje z innymi obiektami oraz mogą oddziaływać na nią różne siły, np.: grawitacja. W silniku wprowadzone jest odgórne zarządzanie równowagą między dokładnością fizyki, a wydajnością gry.

Obiekty znajdujące się w grze podzielone są na statyczne – takie których masa jest nieskończona, oraz dynamiczne, ulegające kolizjom, podatne na zderzenia, odbicia, czy eksplozie. Ilustracja 5.7 przedstawia podział hierarchiczny obiektów.

Ilustracja 5.7. Struktura fizyki w grze.<sup>108</sup>

Obiekty takie jak budynki, czołgi, czy prostopadłościany składające się na ściany, posiadają odpowiadające im jednostki fizyczne, reprezentowane przez pudełka

<sup>106</sup> Źródło własne.

<sup>107</sup> Jest to darmowy silnik fizyki z otwartym oprogramowaniem, stworzony i rozwijany przez R. Nordby.

<sup>108</sup> Źródło własne.

brzegowe. Apteczki oraz pociski odwzorowane są w świecie fizyki jako kule brzegowe. Eksplozja, wynikająca ze zderzenia pocisku z innym obiektem fizycznym, ma charakter kulisty – ekspansyjny, rośnie wraz z czasem. Teren ze względu na swój nierównomierny charakter, reprezentowany jest przez statyczna siatkę trójkątów, dzięki czemu nie oddziałuje na niego siła grawitacji, przy tym zatrzymując od spadku wszystkie znajdujące się nad nim obiekty fizyczne.

### 5.3 Koncepcja modelu Sztucznej Inteligencji

„*Najważniejszy w każdym działaniu jest początek*” [53]

Platon

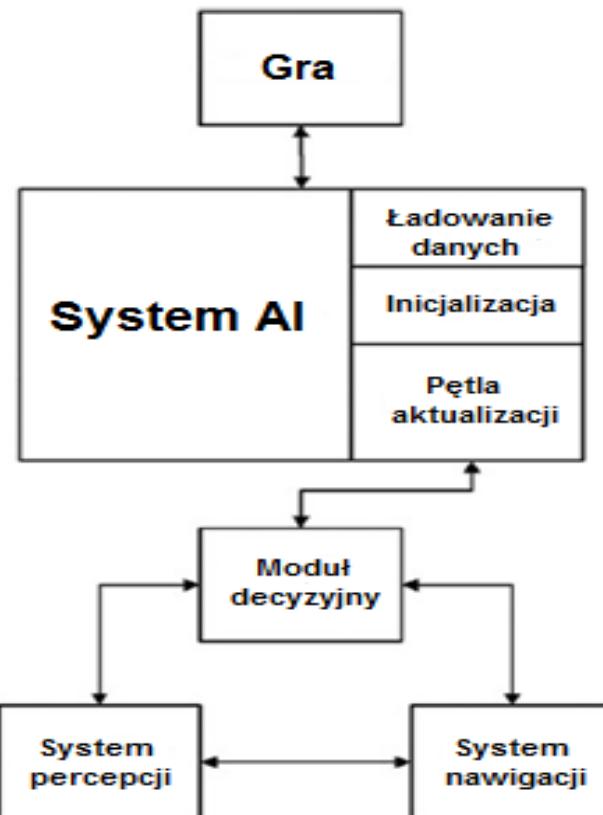
Po zaprojektowaniu środowisko, po którym mogą poruszać się i oddziaływać na siebie graficzne obiekty, kolejna opracowanie modelu Sztucznej Inteligencji, dzięki któremu będą one mogły być inteligentne i autonomiczne. W dziale tym omówione zostaną fundamentalne elementy standardowego silnika AI, wykorzystane przez nas modele zachowania Sztucznej Inteligencji, takie jak automat skończony, automat rozmyty i zachowania sterujące.

- Podstawowe komponenty oraz projekt typowego silnika AI

Sercem każdej liczącej się gry komputerowej na świecie jest Sztuczna Inteligencja dającą odczuć graczowi choćby namiastkę rzeczywistości, w której to pewnie nigdy by się nie znalazł. Podstawą takiego silnika jest jego porządne zaprojektowanie. W wielu książkach znanych autorów powtarza się jedna myśl: by na projektowaniu czasu nie oszczędzać, ponieważ jest to najważniejsza część podczas całej produkcji jakiegokolwiek oprogramowania. Od czego należy zacząć? Wyróżniamy następujące podstawowe kroki przy projektowaniu systemu AI [14]:

- Ustalenie na jakie podsystemy podzielić system AI,
- Ustalenie typu danych wejściowych do systemu,
- Ustalenie typu danych wyjściowych z systemu,
- Ustalenie logiki łączącej dane wejściowe z danymi wyjściowymi.

Ilustracja 5.8 obrazuje podstawowy schemat budowy silnika Sztucznej Inteligencji. Składa się on na trzy główne podsystemy, moduł ładowania danych z gry, moduł inicjalizacyjny – startujący pracę silnika AI oraz moduł aktualizacji – zawierający pętlę uaktualniającą stan, poprzez proces decyzyjny zachodzący w odpowiednich modułach.



Ilustracja 5.8. Podstawowy schemat budowy typowego silnika AI<sup>109</sup>

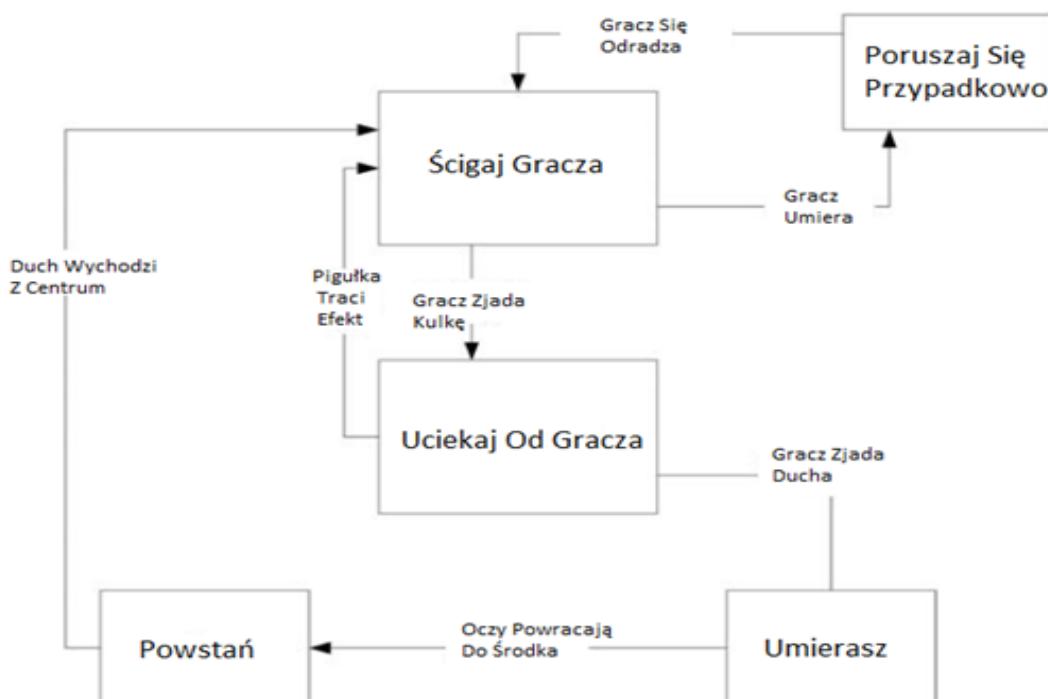
- **System Percepcji** - Jest czymś w rodzaju interfejsu pomiędzy systemem Sztucznej Inteligencji a światem gry. Stanowi bardzo ważnym elementem definiującym jakie dane powinny być przekazywane do systemu, w jakich odstępach czasu i czy powinny zostać zapamiętane. Są to dane wejściowe na podstawie których system będzie podejmował decyzję. System percepcji robota w projekcie zakłada pobieranie takich danych ze świata gry jak położenie przeciwników, przeszkód, apteczki oraz statuy. Pobiera również informacje o stanie zdrowia przeciwnika oraz zniszczeniach statuy. Dodatkowo posiada informacje na temat swojego stanu zdrowia.
- **Moduł decyzyjny** - tutaj znajduje się główna logika systemu łącząca dane wejściowe z wyjściowymi na podstawie których podejmowana jest jakaś akcja. Moduł decyzyjny bazuje na pewnego rodzaju hierarchicznej maszynie stanów, poszczególne stany obliczają swój poziom aktywacji na podstawie danych pobranych ze środowiska. Na podstawie poziomu aktywacji maszyna stanów jest w stanie podjąć decyzję, które stany mogą podjąć działanie. Zadaniem stanów jest obliczenie wektorów sił kierujących robotem. Wektory te są sumowane po czym

<sup>109</sup> Ilustracja pochodzi z książki [22], tłumaczenie własne.

wynik przekazywany jest do systemu mechanizmów wykonawczych, gdzie odpowiednie decyzje przekładane są na poruszanie się robota oraz jego części.

- **System nawigacji** - tutaj znajdują zastosowanie wszelkie algorytmy, które pozwalają na dotarcie z punktu A do punktu B. Systemy te najczęściej bazują na sieciach połączonych ze sobą punktów, które przeszukiwane są za pomocą algorytmu A\*<sup>110</sup>. Chociaż są również inne rozwiązania jak mapy potencjału lub proste reguły unikania obiektów. W projekcie do poruszania robotem wykorzystano zachowania sterujące, system ten jest ściśle powiązany z modułem decyzyjnym, to tutaj każdy stan oblicza swój udział w poruszaniu robotem.
- Inne techniki wykorzystywane przez Sztuczną Inteligencję w grach
  - Automat Skończony

Automaty skończone<sup>111</sup> są bardzo prostą koncepcją, bazującą na stanach w jakich system może się znaleźć oraz połączeń między stanami definiującymi warunki przejść między nimi [14]. Struktura ta jest jedną z najpopularniejszych w świecie Sztucznej Inteligencji gier komputerowych, pozwala programistom w łatwy sposób zorganizować kod.



Ilustracja 5.9. Diagram prostego automatu skończonego czerwonego duszka z opisywanej wcześniej gry Pac-Man.<sup>112</sup>

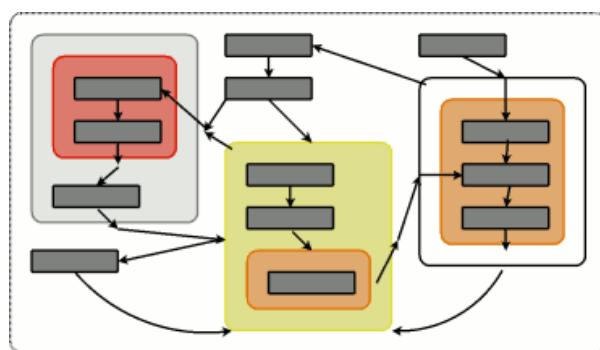
<sup>110</sup> A\* jest to algorytm znajdywania najkrótszej ścieżki pomiędzy dwoma wierzchołkami grafu.

<sup>111</sup> ang. finite state machine (FSM)

<sup>112</sup> Ilustracja pochodzi z książki [22, s. 264], tłumaczenie własne.

Siła automatów skończonych bazuje na prostocie, przez co programiści mają ułatwione zadanie jeśli chodzi o wyszukiwanie błędów oraz modyfikowanie zachowań. Jednak pomimo swojej popularności z automatami związane jest wiele problemów, które opisuje A. J. Champandard w jednym ze swoich artykułów [54], wieszcząc im powolny koniec na rzecz innych technologii takich jak drzewa zachowań. Sposobów implementacji jest kilka, a najpopularniejszą z nich jest automat zakodowany na sztywno w językach takich jak C, czy C++ ponieważ w rezultacie otrzymamy statyczny kod maszynowy. Kolejnym typem jest automat interpretowany, który używa interpretera do ładowania osobnego pliku z opisem stanów oraz przejść między nimi. Oba rozwiązania mają swoje wady i zalety. Pierwsze rozwiązanie jest bardzo szybkie, za to drugie bardziej elastyczne, ułatwiające modyfikacje oraz projektowanie maszyn przez osoby trzecie, nie zmuszając ich do zagłębiania się w szczegóły implementacji. Trzeci sposób jest pewną hybrydą, pomysł polega na stworzeniu automatu na podstawie danych ładowanych z pliku, a następnie skompilowanie go do kodu maszynowego, tak by mógł być uruchomiany jak program [55]. Projekt ze względu na swoją prostotę zakłada zakodowanie maszyny stanów na sztywno.

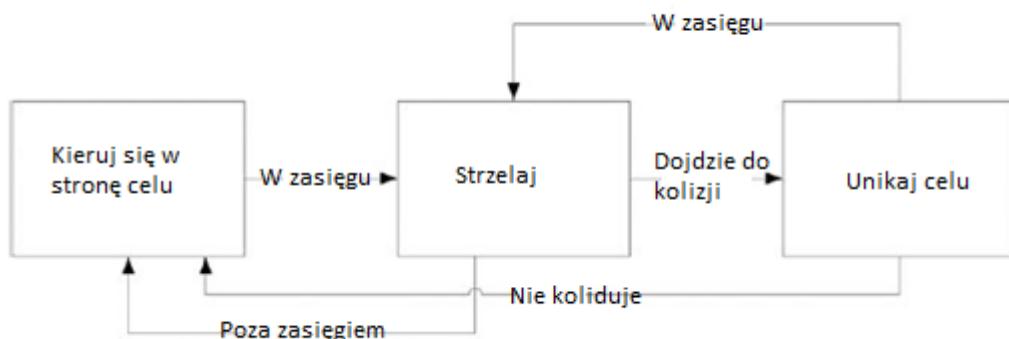
Istnieje również kilka wariacji dotyczących samej architektury automatów np. bazująca na systemie wiadomości oraz zdarzeń jako przejść między stanami by przyśpieszyć obliczenia gdy przejścia nie są wykonywane. Zbyt często jednak jedną z najpopularniejszych jest wariacja hierarchiczna, grupująca zestawy powiązanych ze sobą stanów w podrzędne automaty. W ten sposób możemy odseparować pewne wspólne aspekty różnych stanów, co również skutkuje tym, że kolejne stany możemy dodawać z większą łatwością. Automaty takie znajdują zastosowanie w bardziej skomplikowanych systemach gdzie zarządzanie liczbą stanów oraz przejść między nimi było by bardzo trudne[14, s. 289]. Hierarchiczne maszyny stanów mają wiele wspólnego z drzwiami zachowań opisywanymi przy okazji gry Halo, jednak w przeciwieństwie do stanów w drzewach zachowań podstawowym elementem jest zadanie. Zadania mogą być budulcami bardziej skomplikowanych zadań [23, s. 334]. Natomiast hierarchia automatów skończonych generalizuje tylko poszczególne stany.



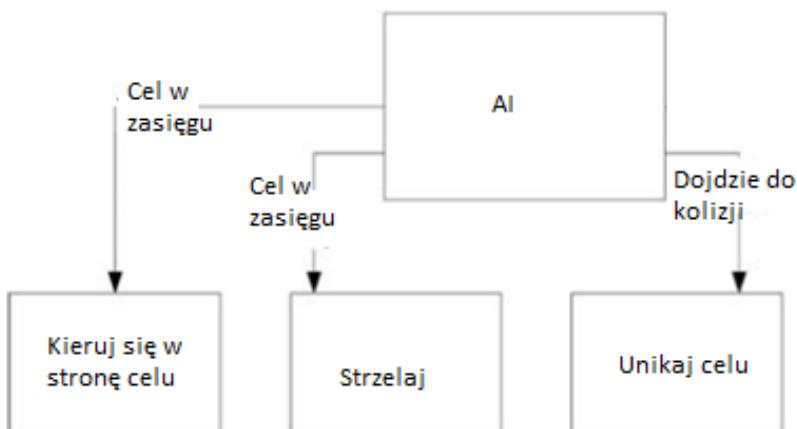
Ilustracja 5.10. Typowy hierarchiczny automat skończony [55].

- Automat Stanów Rozmytych

Rozmytym automatem skończonym może być każdy automat skończony zawierający w sobie jakieś elementy rozmyte jak np. przejścia między stanami używające logiki rozmytej. Automat rozmyty może również posiadać rozmyte stany tzn. system może znajdować się w kilku stanach jednocześnie [23, s. 390]. W swojej książce [14, s. 303] B. Shwab opisuje rozmyty automat skończony jako system zbudowany na pojęciu logiki rozmytej jednak nie reprezentujący takiego systemu. W jego książce również automat rozmyty to taki, który może znaleźć się w większej liczbie stanów, podobnie jak w [20]. Architektura taka znajduje zastosowanie tam, gdzie system może znaleźć się w kilku stanach jednocześnie, gdy stany te reprezentują niezależne od siebie systemy. Przykład obu typów automatu:



Ilustracja 5.11. Przykładowy automat skończony statku kosmicznego<sup>113</sup>.



Ilustracja 5.12. Przykładowy rozmyty automat skończony tego samego statku kosmicznego.<sup>114</sup>

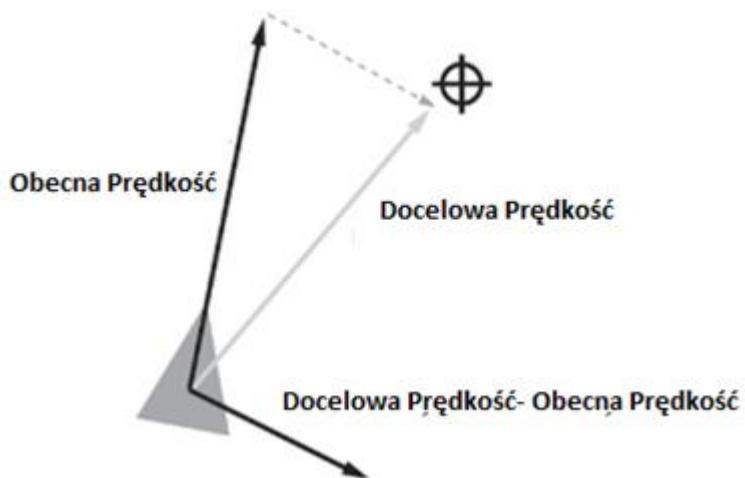
<sup>113</sup> Ilustracja pochodzi z książki [14, s. 307], tłumaczenie własne.

<sup>114</sup> Ilustracja pochodzi z książki [14, s. 315], tłumaczenie własne.

- Zachowania Sterujące

Zestaw algorytmów należących do zachowań sterujących<sup>115</sup> to tak naprawdę zestaw prostych reguł skupiających się na poruszaniu obiekty. Reprezentują niskopoziomowe podejmowanie decyzji, skupiają się tylko na tym w którą stronę agent powinien się poruszać i z jaką prędkością. Przedstawione zostaną tutaj tylko wybrane zachowania wykorzystane w projekcie.

**Podążanie** - reguła pozwalająca znaleźć wektor siły, która musi zadziałać by nakierować obiekt na cel.



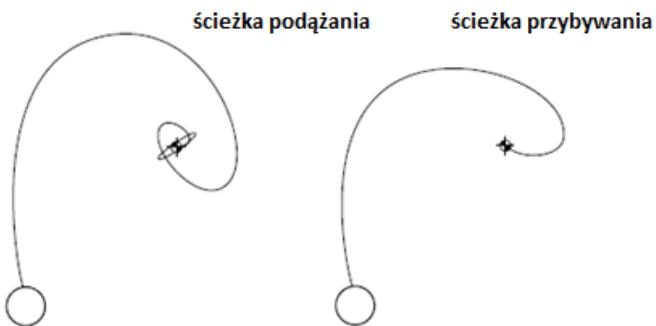
Ilustracja 5.13. Obliczanie poszukiwanego wektora.<sup>116</sup>

**Ucieczka** - przeciwnieństwo podążania, zamiast poruszać się w kierunku celu chcemy się od niego oddalić. Różnica polega na tym, że w obliczeniu wektora siły uwzględniamy wektor przeciwny do wektora kierunku ruchu.

**Przybycie** - reguła podążania zakłada poruszanie się w kierunku celu z pełną prędkością, nie sprawia to problemów, gdy cel ucieka lub sterowany obiekt może natychmiast zmienić kierunek ruchu, jednak w większości wypadków poruszający się obiekt skręca również z jakąś szybkością przez co osiągnięcie jakiegoś punktu na mapie mogło by powodować oscylacje wokół tego punktu. Reguła przybycia zakłada zmniejszanie prędkości w raz ze zmniejszaniem się odległości do celu, co nie powoduje oscylacji oraz daje miły dla oka efekt.

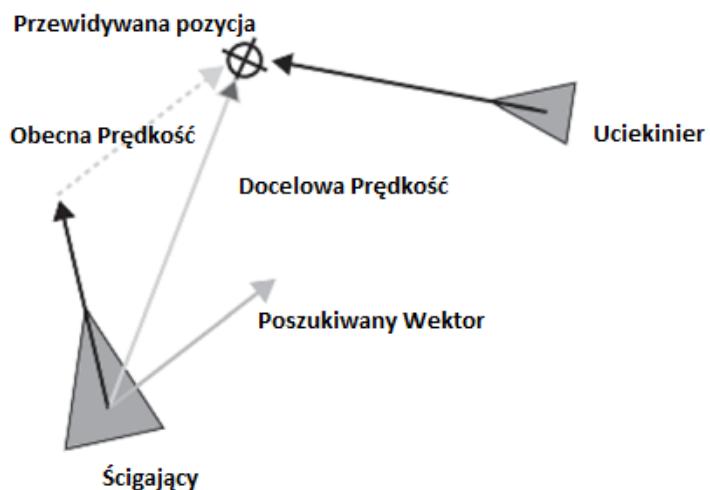
<sup>115</sup> Opracowano na podstawie książek autorów M. Buckland [16], I. Millington, J. Funge. [20]

<sup>116</sup> Ilustracja z książki [16], tłumaczenie własne.



Ilustracja 5.14. Porównanie trasy zachowań podążania oraz przybycia, gdzie celem jest obiekt nieruchomy.<sup>117</sup>

**Pościg** - reguła przydatna gdy agent chce złapać poruszający się obiekt lub zablokować przeciwnikowi drogę. Agent stara się przewidzieć punkt do, którego zmierza obiekt zainteresowania i dążyć do tego punktu zamiast bezmyślnie w kierunku jego obecnego położenia.



Ilustracja 5.15. Obliczanie poszukiwanego wektora nakierowującego agenta na przewidziany punkt docelowy<sup>118</sup>.

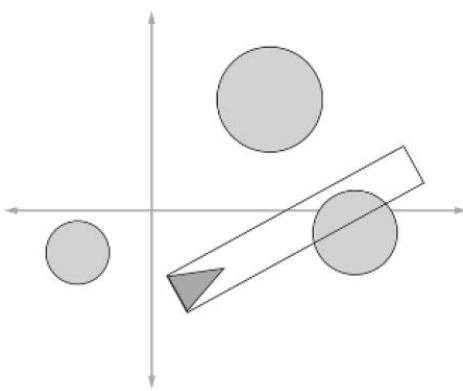
**Unikanie** - Jest bardzo podobne do pościgu, jednak tym razem agent chce uniknąć punktu w którym znajdzie się inny agent lub obiekt.

**Błędzenie** - zadaniem algorytmu jest tworzenie wrażenia wałsania się agenta po środowisku bez konkretnego celu. Nie ma jednego algorytmu, który osiągnął by ten cel. Podejścia są różne, najprostsze to generowanie przypadkowych wektorów co określoną ilość sekund. Ale rozwiązanie to jest naiwne ponieważ może powodować zbyt drastyczne zmiany kierunków.

<sup>117</sup> Ilustracja pochodzi z książki [20], tłumaczenie własne.

<sup>118</sup> Ilustracja z książki [16], tłumaczenie własne.

**Omijanie przeszkód** - jest algorytmem sterującym agentem tak by płynnie unikał przeszkód stojących mu na drodze. Osiągnąć to można za pomocą obszaru, który jest figurą płaską lub przestrzenną przyczepioną do agenta umożliwiającą wykrycie obiektów znajdujących się przed nim.

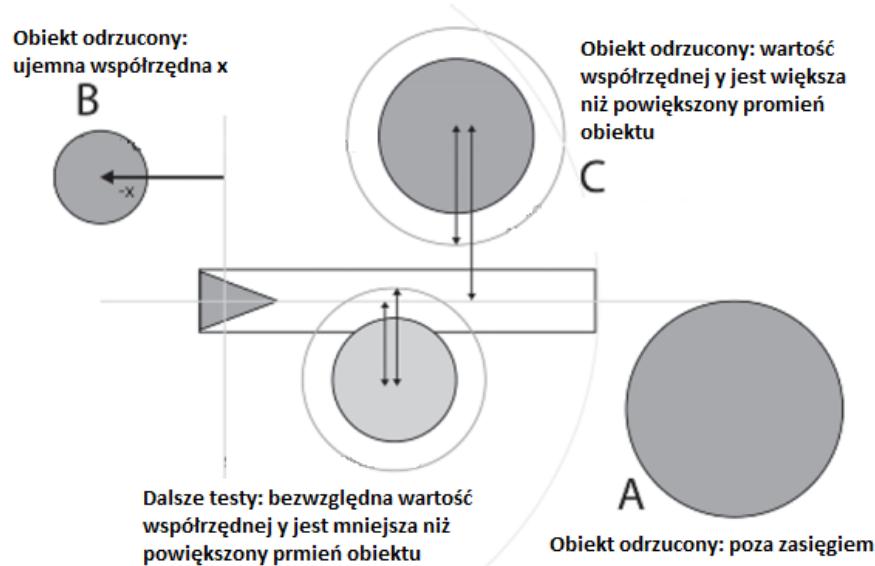


Ilustracja 5.16. Obszar detekcji przecinający obiekt w kształcie koła<sup>119</sup>.

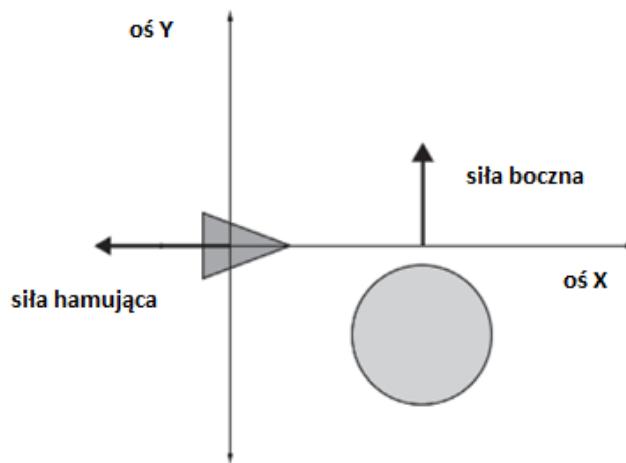
Aby znaleźć najbliższy punkt przecięcia z przeszkodą należy:

- A)** Sprawdzić wszystkie obiekty w grze, czy znajdują się w zasięgu naszego obszaru detekcji kolizji i odrzucić z dalszej części algorytmu te, które są poza zasięgiem.
- B)** Następnie należy pozostałe obiekty przetransformować z globalnego układu współrzędnych do lokalnego układu współrzędnych agenta tj. układu gdzie agent znajduje się w punkcie (0,0) skierowany najczęściej w kierunku dodatniej osi x w świecie 2D lub ujemnej osi z w świecie 3D. Następnie w łatwy sposób można wyrzucić wszystkie obiekty znajdujące się poza agentem.
- C)** Ostatnim krokiem jest porównanie lokalnej wartości współrzędnej y dla świata 2D oraz x dla świata 3D promienia przeszkody. W tym celu należy promień przeszkody rozszerzyć o połowę szerokości obszaru detekcji agenta w celu sprawdzenia czy obiekt znajduje się na obszarze detekcji. Jeżeli wartość współrzędnej obiektu jest mniejsza od promienia obszaru przeszkody wtedy obiekt przecina obszar detekcji agenta. Jeżeli jest większa, to obiekt nie stoi mu na drodze i można go pominąć.
- D)** Obecnie mamy już tylko obiekty, które przecinają drogę agentowi. Teraz należy znaleźć punkty przecięcia z obiektami by znaleźć ten najbliższy. Do tego wystarczy znaleźć miejsca zerowe funkcji okręgu otaczającego obiekt o rozszerzonym promieniu z poprzedniego kroku.

<sup>119</sup> Ilustracja z książki [16], tłumaczenie własne.

Ilustracja 5.17. Kroki A, B i C<sup>120</sup>.

**E)** Ostatnim krokiem jest wyliczenie siły sterującej naszym pojazdem. Role odgrywają tutaj 2 siły. Pierwsza wzrasta wraz ze zbliżaniem się agenta do obiektu i jest skierowana przeciwnie do kierunku jego ruchu, natomiast druga jest skierowana prostopadle do kierunku jego ruchu i ma za zadanie odeprzeć pojazd od przeszkody.

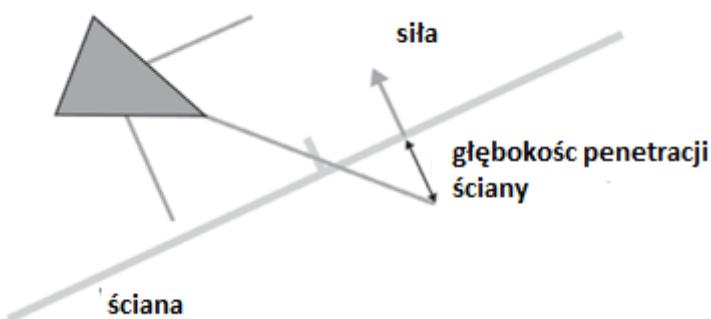
Ilustracja 5.18. Składowe poszukiwanego wektora siły sterującej<sup>121</sup>.

Podany algorytm pochodzi z książki [44] i zakłada detekcję w 2 wymiarach, aby zastosować go w detekcji w 3 wymiarach autor sugeruje wykorzystanie sfer przy opisie obiektów oraz cylindra jako obszaru detekcji co ułatwia obliczenia. Jak można zauważyć algorytm zakłada, że przeszkody posiadają swój promień tzn. w świecie 3D każdy obiekt powinien mieć zdefiniowaną sferę otaczającą go. Jest to jak najbardziej słuszne ponieważ

<sup>120</sup> Ilustracja z książki [16], tłumaczenie własne.<sup>121</sup> Ilustracja z książki [16], tłumaczenie własne.

wszelkiego rodzaju testy na sferach są prostsze, przez co nie obciążają procesora, dlatego zawsze gdy tylko się da, należy wszelkiego rodzaju obiekty przybliżać sferami, lecz nie zawsze jest to możliwe.

**Unikanie ścian** - zadaniem algorytmu jest generowanie sił przeciwdziałającym zderzeniom z wszelkiego rodzaju ścianami (prostokątnymi segmentami lub w świecie 3D segmentami prostopadłościianów), które posiadają wektory normalne zwrocone w kierunku ich skierowania. Różnica w porównaniu do poprzedniej sekcji wynika z tego, że murów nie da się opisać za pomocą sfer ani okręgów, czasem nie jest możliwe nawet ich ominięcie, dlatego też potrzebny jest inny algorytm.



Ilustracja 5.19. Unikanie ściany<sup>122</sup>

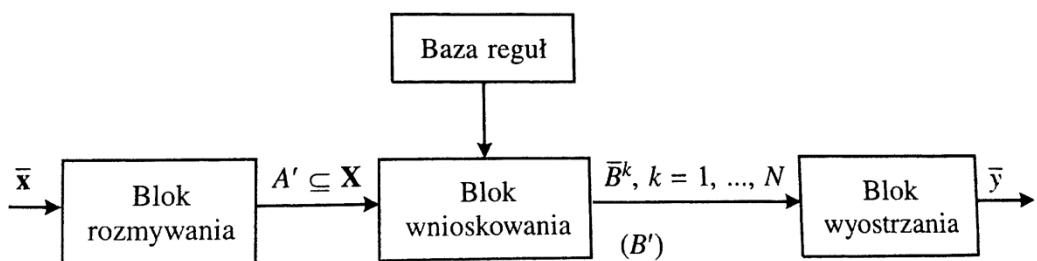
Pomysł polega na przyczepieniu do agenta trzech czujników, które są w stanie przedwcześnie wykryć kolizję z obiektem. Gdy już taka kolizja zostanie wykryta generowana jest siła w kierunku wektora normalnego ściany o długości proporcjonalnej do głębokości penetracji ściany przez czujnik.

#### 5.4 Rozmyte systemy wnioskujące

Teoria zbiorów rozmytych jest wygodnym narzędziem, które znajduje zastosowanie m.in. przy wspomaganiu podejmowania decyzji, klasyfikacji, sterowaniu urządzeniami jak również skomplikowanymi procesami przemysłowymi. Jedynie czego potrzeba to zdefiniowania reguł typu "Jeżeli...To..." by wymodelować przybliżone zachowanie systemu. Reguły te stanowią bazę wiedzy systemu, są połączeniem między lingwistyczną informacją, a matematyką definiującą jego zachowanie, które w głównej mierze bazuje na wiedzy pozyskanej z doświadczeń ludzi. Zdolności ludzkie pozwalają na kontrolowanie skomplikowanych procesów takich jak np. prowadzenie samochodu w

<sup>122</sup> Ilustracja z książki [16], tłumaczenie własne.

miejscu o natężonym ruchu, co nie jest łatwe do osiągnięcia przez konwencjonalne systemy kontroli bez stworzenia odpowiedniego modelu rozważanego procesu. Jednak takie modele mogą być bardzo skomplikowane, wymagać dużo czasu do ich stworzenia lub solidnej wiedzy teoretycznej. Projekt systemu wnioskowania bazuje na wiedzy empirycznej dotyczącej danego zagadnienia zamiast na dogłębnej analizie problemu, ponieważ jeżeli człowiek jest uwikłany w kontrolowanie jakiegoś procesu, ma dostęp do dużej wiedzy będącej w formie prostych procedur. Wiedza taka może bezpośrednio zostać rzutowana na formę językową. Typowy schemat systemu wnioskującego składa się z czterech głównych elementów: bazy reguł, bloku rozmywania, bloku wnioskowania oraz bloku wyostrzania. Zawarte w tym materiale wzory, definicje oraz ilustracje pochodzą z rozdziału 4 książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”.



Ilustracja 5.20. Rozmyty system wnioskujący.<sup>123</sup>

Baza reguł nazywana również modelem lingwistycznym, składa się na nią zbiór reguł typu "Jeżeli...To..." np. "Jeżeli temperatura jest zbyt wysoka To zmniejsz odpowiednio podgrzewanie"

$$R^{(k)}: \text{JEŻELI } x_1 \text{ jest } A_1^k \text{ I } x_2 \text{ jest } A_2^k \text{ I } \dots \text{ I } x_n \text{ jest } A_n^k \text{ TO } y_1 \text{ jest } B_1^k \text{ I } \dots \text{ I } y_m \text{ jest } B_m^k, \quad (79)$$

gdzie:

$R^{(k)}$ ,  $k = 1, \dots, N$  - oznacza daną regułę,

N - liczba reguł,

$X_i, i = 1, \dots, n$  - przestrzeń zmiennych wejściowych,

$Y_j, j = 1, \dots, m$  - przestrzeń zmiennych wyjściowych,

$A_i^k$  - zbiory rozmyte takie, że  $A_i^k \subseteq X_i \subset R, i = 1, \dots, n$ ,

$B_j^k$  - zbiory rozmyte takie, że  $B_j^k \subseteq Y_j \subset R, i = 1, \dots, m$ ,

$x_1, x_2, \dots, x_n$  - zmienne wejściowe modelu lingwistycznego, przy czym zachodzi:

$$[x_1, x_2, \dots, x_n]^T = x \in X_1 \times X_2 \times \dots \times X_n \quad (80)$$

$y_1, y_2, \dots, y_m$  - zmienne wyjściowe modelu lingwistycznego, przy czym zachodzi:

$$[y_1, y_2, \dots, y_m]^T = y \in Y_1 \times Y_2 \times \dots \times Y_m \quad (81)$$

<sup>123</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

„Każda reguła składa się z części JEŻELI, nazywanej poprzednikiem (ang. antecedent), oraz części TO zwanej następnikiem (ang. consequent). Poprzednik zawiera zbiór warunków, natomiast następnik zawiera wniosek.” [13] Zmienne  $x$  oraz  $y$  mogą przyjmować wartości słowne oraz liczbowe. Przyjęte zostanie również, że poszczególne reguły  $R^{(k)}$  są powiązane ze sobą za pomocą operatora logicznego "lub", oraz wyjścia  $y_1, y_2, \dots, y_m$  są niezależne wzajemnie, przez co bez utraty ogólności rozpatrywane reguły będą posiadały skalarne wyjście:

$$R^{(k)}: \text{JEŻELI } x_1 \text{ jest } A_1^k \text{ I } x_2 \text{ jest } A_2^k \text{ I } \dots \text{ I } x_n \text{ jest } A_n^k \\ \text{TO } y \text{ jest } B^k. \quad (82)$$

gdzie:

$$B^k \subseteq Y \subset R, \quad k = 1, \dots, N.$$

Stosując poniższe oznaczenia:

$$\begin{aligned} X &= X_1 \times X_2 \times \dots \times X_n, \\ Y &= Y_1 \times Y_2 \times \dots \times Y_m. \end{aligned} \quad (83)$$

Regułę (82) można zapisać jako rozmytą implikację:

$$R^{(k)}: A^k \rightarrow B^k, \quad k = 1, \dots, N. \quad (84)$$

Regułę  $R^{(k)}$  można również interpretować jako rozmytą relację, gdzie  $R^{(k)} \subseteq X \times Y$  oraz funkcję przynależności określa się jako:

$$\mu_{R^{(k)}}(x, y) = \mu_{A^k \rightarrow B^k}(x, y). \quad (85)$$

Baza reguł jest kluczowym elementem przy projektowaniu sterowników, tutaj należy podjąć decyzję oraz rozstrzygnąć o formie reguł, ich liczbę, czy są one sprzeczne i czy zachodzą między nimi interakcje. Model systemu wnioskowania bazujący na regułach typu (79) jest tzw. modelem Mamdaniego. W takim przypadku poprzedniki oraz następnik są zbiorami rozmytymi. Innym ciekawym modelem jest Takagi-Sugeno, gdzie poprzednikami wciąż są zbiory rozmyte jednak następniem jest już wartość nie rozmyta. W dalszej części pracy rozważaniom będzie poddany model Mamdaniego.

„Blok Rozmywania” jest miejscem gdzie konkretne wartości podane na wejście systemu zostają przekonwertowane na formę rozmytą tzn. wartość ta zostaje odwzorowana w zbiór rozmyty by mogła zostać poddana dalszemu procesowi. W zagadnieniach sterowania najczęściej stosuje się operację rozmywania typu singleton (8) zdefiniowaną jako:

$$\mu_{A'}(x) = \delta(x - \bar{x}) = \begin{cases} 1, & \text{jeżeli } x = \bar{x}, \\ 0, & \text{jeżeli } x \neq \bar{x}. \end{cases} \quad (86)$$

Na wejściu „Blok Wnioskowania” jest już zbiór rozmyty  $A'$ , na wyjściu wynikiem może być jeden zbiór rozmyty  $B'$  lub więcej zbiorów rozmytych  $\bar{B}^{(k)}$ , obu przypadkom odpowiadają różne metody wyostrzania. Zbiór rozmyty  $\bar{B}^{(k)}$  jest określony przez złożenie zbioru rozmytego  $A'$  i relacji  $R^{(k)}$ .

$$\bar{B}^{(k)} = A' \circ (A^k \rightarrow B^k), k = 1, \dots, N. \quad (87)$$

Natomiast funkcja przynależności zbioru  $\bar{B}^{(k)}$  na podstawie definicji złożenia zbioru rozmytego oraz relacji rozmytej przyjmuje postać:

$$\mu_{\bar{B}^k}(y) = \sup_{x \in X} [\mu_{A'}(x)^T * \mu_{A^k \rightarrow B^k}(x, y)]. \quad (88)$$

Konkretna postać funkcji  $\mu_{\bar{B}^k}(y)$  zależy od trzech czynników:

1. Przyjętej t-normy,
2. Przyjętej reguły wnioskowania,
3. Sposobu zdefiniowania iloczynu kartezjańskiego zbiorów rozmytych.

W przypadku użycia metody singleton przy operacji rozmywania wzór (88) redukuje się do postaci:

$$\mu_{\bar{B}^k}(y) = \sup_{x \in X} [\mu_{A^k \rightarrow B^k}(\bar{x}, y)]. \quad (89)$$

Dla przykładu<sup>124</sup>, jeżeli  $n=2$ , t-norma jest typu minimum, rozmyte wnioskowanie definiuje reguła typu minimum oraz iloczyn kartezjański zbiorów rozmytych określa wzór (40) to  $\mu_{\bar{B}^k}(y)$  przybiera postać:

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \sup_{x \in X} [\min[\mu_{A'}(x), \mu_{A^k \rightarrow B^k}(x, y)]] \\ &= \sup_{x \in X} \{\min[\mu_{A'}(x), \min[\mu_{A^k}(x), \mu_{B^k}(y)]]\} \\ &= \sup_{x_1 \in X_1, x_2 \in X_2} \{\min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2), \mu_{A^k_1}(x_1), \mu_{A^k_2}(x_2), \mu_{B^k}(y)]\}. \end{aligned} \quad (90)$$

ponieważ:

$$\mu_{A^k}(x) = \mu_{A_1^k \times A_2^k}(x_1, x_2) = \min[\mu_{A_1^k}(x_1), \mu_{A_2^k}(x_2)] \quad (91)$$

$$\mu_{A'}(x) = \mu_{A'_1 \times A'_2}(x_1, x_2) = \min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)]. \quad (92)$$

Analogicznie jeżeli t-norma jest typu iloczyn, rozmyte wnioskowanie definiuje reguła typu iloczyn oraz iloczyn kartezjańskich zbiorów rozmytych określa wzór (41), to równanie przybiera postać:

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \sup_{x \in X} [\mu_{A'}(x) \cdot \mu_{A^k \rightarrow B^k}(x, y)] \\ &= \sup_{x \in X} \{\mu_{A'}(x) \cdot \mu_{A^k}(x) \cdot \mu_{B^k}(y)\} \\ &= \sup_{x_1 \in X_1, x_2 \in X_2} \{\mu_{A'_1}(x_1) \cdot \mu_{A'_2}(x_2) \cdot \mu_{A^k_1}(x_1) \cdot \mu_{A^k_2}(x_2) \cdot \mu_{B^k}(y)\}. \end{aligned} \quad (93)$$

Z powyższych równań dostajemy k zbiorów rozmytych, aby na wyjściu bloku otrzymać jeden zbiór rozmyty  $B'$  należy zsumować otrzymane zbiory zgodnie z definicją sumy zbiorów rozmytych.

---

<sup>124</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

$$B' = \bigcup_{k=1}^N A' \circ R^{(k)} = \bigcup_{k=1}^N A' \circ (A^k \rightarrow B^k). \quad (94)$$

funkcja przynależności zbioru rozmytego  $B'$  dana jest wzorem:

$$\mu_{B'}(y) = S_{k=1}^N \mu_{\bar{B}^k}(y), \quad (95)$$

gdzie funkcja  $\mu_{\bar{B}^k}(y)$  jest określona wzorem (88).

Dla przykładu<sup>125</sup>, dany jest rozmyty system wnioskujący z następującą bazą reguł:

$$R^{(1)}: JEŻELI x_1 \text{ jest } A_1^1 \text{ I } x_2 \text{ jest } A_2^1 \text{ TO } y \text{ jest } B^1, \quad (96)$$

$$R^{(2)}: JEŻELI x_1 \text{ jest } A_1^2 \text{ I } x_2 \text{ jest } A_2^2 \text{ TO } y \text{ jest } B^2, \quad (97)$$

Na wejście systemu został podany sygnał  $\bar{x} = [\bar{x}_1, \bar{x}_2]^T$ . Sygnał ten został poddany operacji rozmywania typu singleton w wyniku, której otrzymano zbiory rozmyte  $A'_1$  oraz  $A'_2$  gdzie:

$$\mu_{A'_1}(x_1) = \delta(x_1 - \bar{x}_1), \mu_{A'_2}(x_2) = \delta(x_2 - \bar{x}_2). \quad (98)$$

Do wyznaczenia sygnału wyjściowego bloku wnioskowania jako t-normę zostanie przyjęta operacja minimum. Na podstawie wzoru (88) mamy:

$$\mu_{\bar{B}^k}(y) = \sup_{x_1 \in X_1, x_2 \in X_2} [\min[\mu_{A'_1 \times A'_2}(x_1, x_2), \mu_{R^{(k)}}(x_1, x_2, y)]]. \quad (99)$$

Ponadto zostanie przyjęte, że:

$$\begin{aligned} \mu_{A'_1 \times A'_2}(x_1, x_2) &= \min[\mu_{A'_1}(x_1), \mu_{A'_2}(x_2)] \\ &= \min[\delta(x_1 - \bar{x}_1), \delta(x_2 - \bar{x}_2)]. \end{aligned} \quad (100)$$

Dlatego wzór na  $\mu_{\bar{B}^k}(y)$  przyjmuje postać:

$$\begin{aligned} \mu_{\bar{B}^k}(y) &= \sup_{x_1 \in X_1, x_2 \in X_2} [\min[\delta(x_1 - \bar{x}_1), \delta(x_2 - \bar{x}_2)], \mu_{R^{(k)}}(x_1, x_2, y)] \\ &= \mu_{R^{(k)}}(\bar{x}_1, \bar{x}_2, y) \end{aligned} \quad (101)$$

gdzie:

$$\mu_{R^{(k)}}(\bar{x}_1, \bar{x}_2, y) = \mu_{A_1^k \times A_2^k \rightarrow B^k}(\bar{x}_1, \bar{x}_2, y). \quad (102)$$

Przy zastosowaniu reguły typu minimum Mamdaniego otrzymujemy:

$$\mu_{A_1^k \times A_2^k \rightarrow B^k}(\bar{x}_1, \bar{x}_2, y) = \min [\mu_{A_1^k \times A_2^k}(\bar{x}_1, \bar{x}_2), \mu_{B^k}(y)]. \quad (103)$$

Ponadto:

$$\mu_{A_1^k \times A_2^k}(\bar{x}_1, \bar{x}_2) = \min [\mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2)]. \quad (104)$$

---

<sup>125</sup> Przykład pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

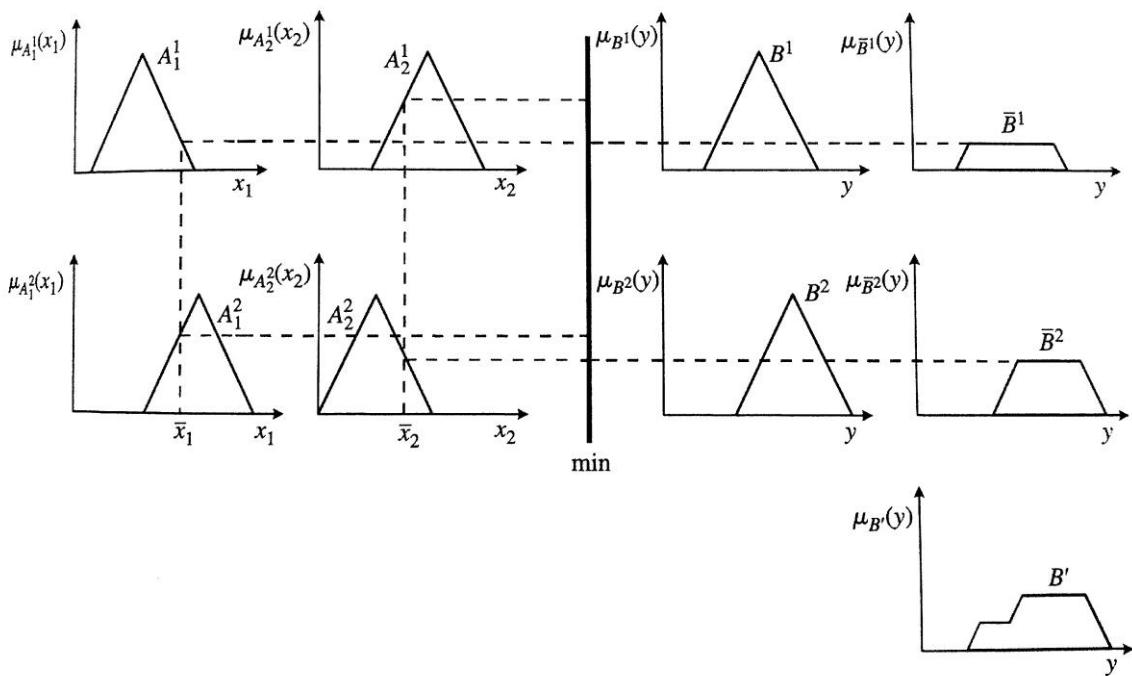
W rezultacie:

$$\begin{aligned}\mu_{\bar{B}^k}(y) &= \min \left\{ \min \left[ \mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2) \right], \mu_{B^k}(y) \right\} \\ &= \min \left[ \mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2), \mu_{B^k}(y) \right]\end{aligned}\quad (105)$$

oraz

$$\mu_{B'}(y) = \max_{k=1,2} \{ \min \left[ \mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2), \mu_{B^k}(y) \right] \}. \quad (106)$$

Poniższa ilustracja przedstawia graficzną interpretację powyższego przykładu rozmytego wnioskowania.



Ilustracja 5.21. Rozmyte wnioskowanie- reguły Mamdaniego.<sup>126</sup>

Analogicznie jeżeli zamiast reguły Mamdaniego zostanie zastosowana reguła typu iloczyn (Larsena), równanie przyjmie postać:

$$\mu_{A_1^k \times A_2^k \rightarrow B^k}(\bar{x}_1, \bar{x}_2, y) = \mu_{A_1^k \times A_2^k}(\bar{x}_1, \bar{x}_2) \cdot \mu_{B^k}(y). \quad (107)$$

W wyniku połączenia obu reguł funkcja przynależności zbioru  $B'$  przyjmie postać:

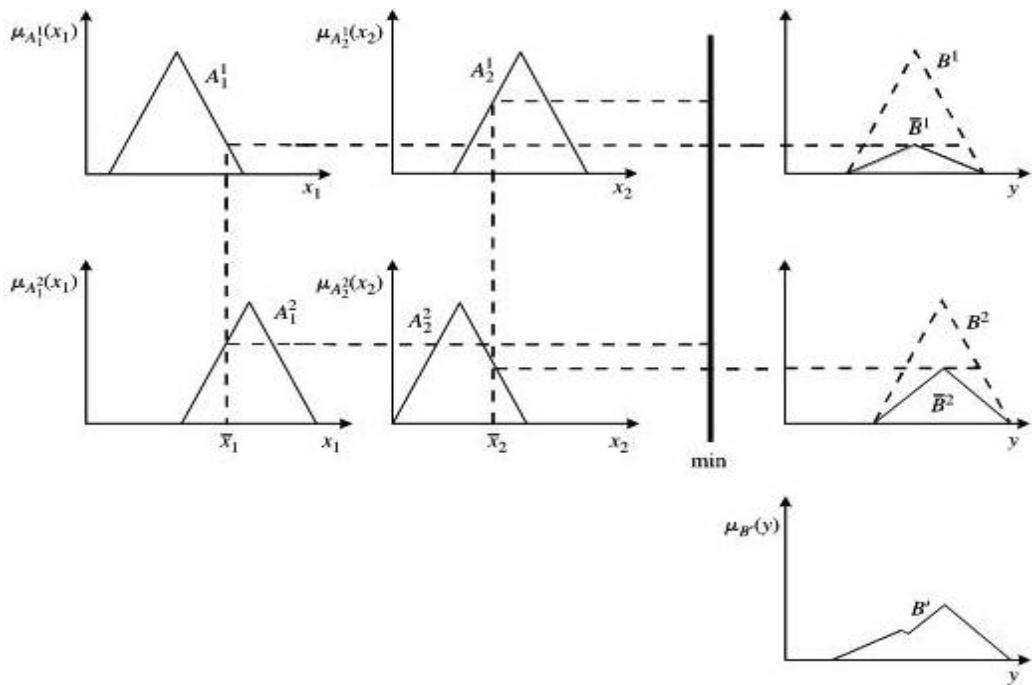
$$\mu_{B'}(y) = \max_{k=1,2} \{ \mu_{B^k}(y) \cdot \min \left[ \mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2) \right] \}, \quad (108)$$

gdzie:

$$\mu_{\bar{B}^k}(y) = \mu_{B^k}(y) \cdot \min \left[ \mu_{A_1^k}(\bar{x}_1), \mu_{A_2^k}(\bar{x}_2) \right]. \quad (109)$$

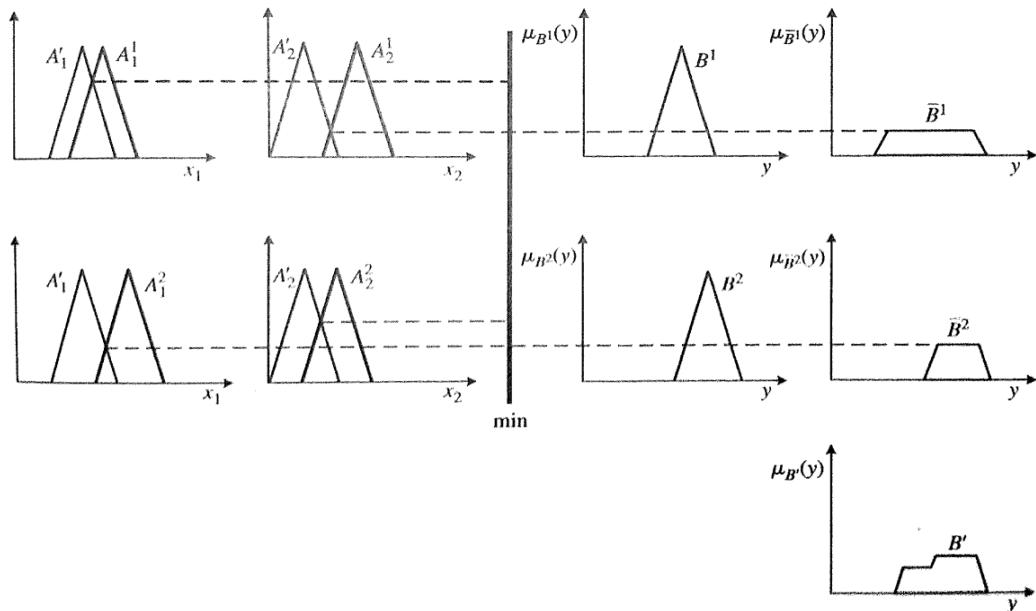
<sup>126</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

Poniższa ilustracja przedstawia graficzną interpretację powyższego przykładu rozmytego wnioskowania dla reguły Larsena.



Ilustracja 5.22. Rozmyte wnioskowanie – reguły Larsena.<sup>127</sup>

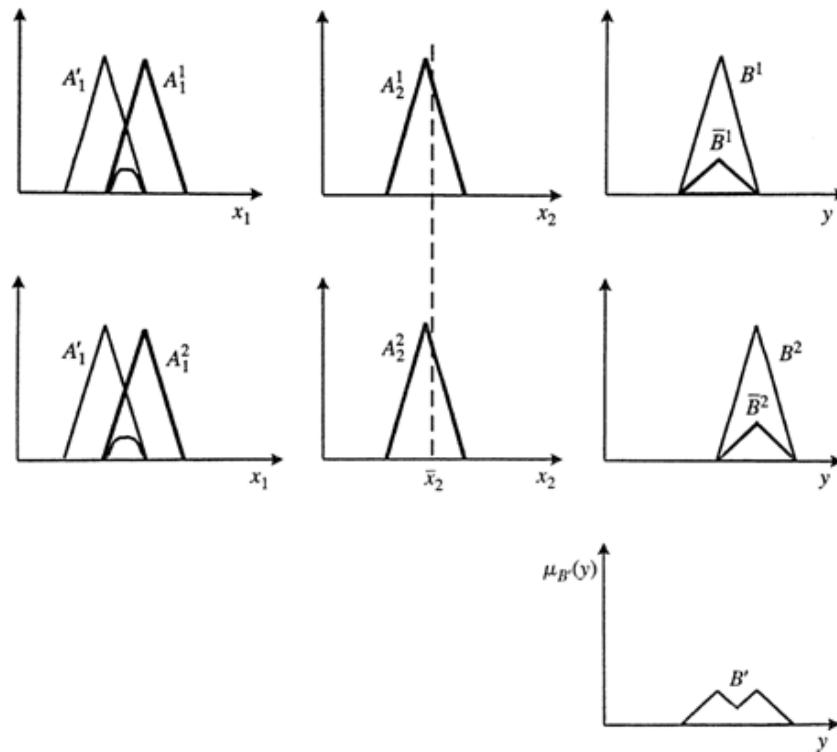
W przypadku gdy na wejściu do bloku wnioskowania mamy dwa zbiory rozmyte nie będące typu singleton, szukamy wysokości zbioru będącego częścią wspólną sygnału wejściowego oraz poprzedników reguły. Ilustracja 5.22 obrazuje ten sposób:



Ilustracja 5.23. Rozmyte wnioskowanie – część wspólna sygnału wejściowego.<sup>128</sup>

<sup>127</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

Ilustracja 5.24 przedstawia przypadek, gdzie jako operację t-normy oraz jako operację wnioskowania został przyjęty iloczyn:



Ilustracja 5.24. Rozmyte wnioskowanie – t-norma i iloczny.<sup>129</sup>

Wyjściem „*Bloku Wystrzania*” może być N zbiorów rozmytych  $\bar{B}^k$  z funkcjami przynależności  $\mu_{\bar{B}^k}(y)$  dla  $k = 1, 2, \dots, N$  lub jeden zbiór rozmyty  $B'$  z funkcją przynależności  $\mu_{B'}(y)$ . W zależności od wyjścia stosowane są różne typy operacji wystrzania. Jeżeli na wyjściu otrzymamy N zbiorów rozmytych, wielkość  $\bar{y}$  oblicza się następująco:

1) Metoda „*center average defuzzification*”. Wartość  $\bar{y}$  wyznacza wzór:

$$\bar{y} = \frac{\sum_{k=1}^N \mu_{\bar{B}^k}(\bar{y}^k) \bar{y}^k}{\sum_{k=1}^N \mu_{\bar{B}^k}(\bar{y}^k)}. \quad (110)$$

gdzie

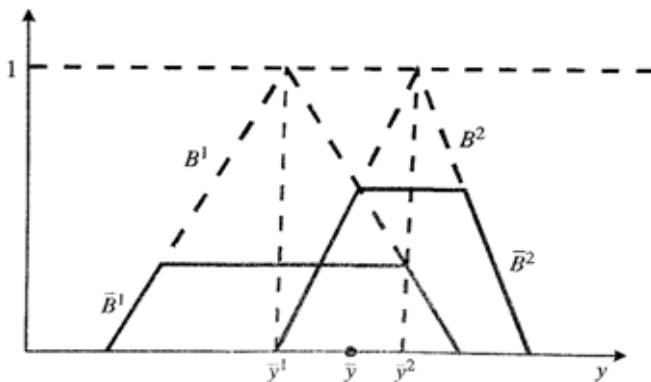
$\bar{y}^k$  - jest środkiem (ang. center) zbioru rozmytego  $B^k$ , czyli punktem, gdzie funkcja  $F$  przyjmuje wartość maksimum, tzn.

$$\mu_{B^k}(\bar{y}^k) = \max_y \mu_{B^k}(y). \quad (111)$$

<sup>128</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

<sup>129</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

Wartość  $\bar{y}$  nie zależy od kształtu funkcji przynależności  $\mu_{B^k}(y)$ . Ilustracja 5.25 obrazuje działanie metody:



Ilustracja 5.25. Rozmyte wyostrzanie – centrum średniej defuzyfikacji.<sup>130</sup>

2) Metoda „center of sums defuzzification”. Wartość  $\bar{y}$  wyznacza wzór:

$$\bar{y} = \frac{\int_Y y \sum_{k=1}^N \mu_{\bar{B}^k}(y)}{\int_Y \sum_{k=1}^N \mu_{\bar{B}^k}(y)}. \quad (112)$$

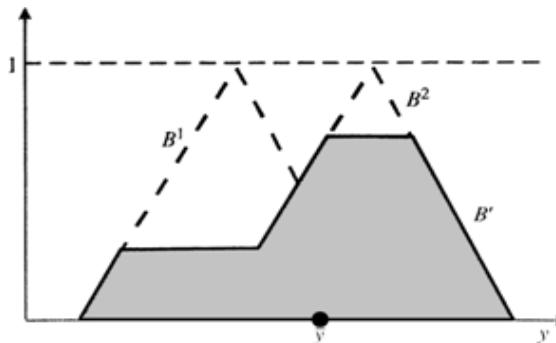
Jeżeli na wyjściu otrzymano jeden zbiór rozmyty  $B'$  to wartość  $\bar{y}$  można wyznaczyć poprzez następującą metodę

3) Metoda „środka ciężkości” (ang. „center of gravity method” lub „center of area method”). Wartość  $\bar{y}$  jest obliczana jako środek ciężkości funkcji przynależności  $\mu_{B'}(y)$  (Ilustracja 5.26), należy również założyć, że całki w poniższym wzorze istnieją.

$$\bar{y} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy} = \frac{\int_Y y S_{k=1}^N \mu_{\bar{B}^k}(y)}{\int_Y S_{k=1}^N \mu_{\bar{B}^k}(y)}. \quad (113)$$

W przypadku otrzymania wyniku dyskretnego, równanie przyjmuje postać:

$$\bar{y} = \frac{\sum_{k=1}^N \mu_{B'}(\bar{y}^k) \bar{y}^k}{\sum_{k=1}^N \mu_{B'}(\bar{y}^k)}. \quad (114)$$



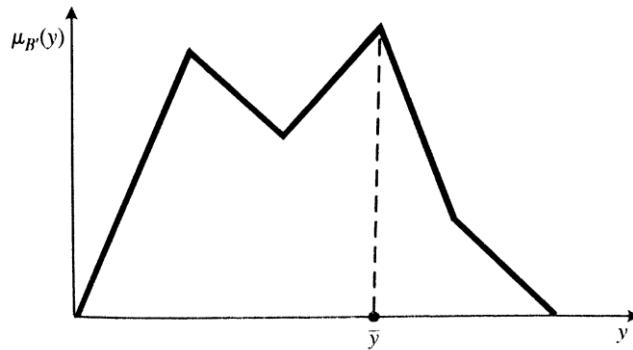
Ilustracja 5.26. Rozmyte wyostrzanie – środek ciężkości.<sup>131</sup>

<sup>130</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „Metody i techniki sztucznej inteligencji”. [13]

- 4) Ostatnią metodą jest „*maksimum funkcji przynależności*”. W metodzie tej kształt funkcji przynależności nie jest uwzględniony

$$\mu_{B'}(\bar{y}) = \sup_{y \in Y} \mu_{B'}(y) \quad (115)$$

gdzie  $\mu_{B'}(y)$  jest funkcją unimodalną. Ilustracja 5.27 obrazuje działanie metody:



Ilustracja 5.27. Rozmyte wyostrzanie – maksimum funkcji przynależności.<sup>132</sup>

<sup>131</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „*Metody i techniki sztucznej inteligencji*”. [13]

<sup>132</sup> Ilustracja pochodzi z rozdziału 4, książki L. Rutkowskiego „*Metody i techniki sztucznej inteligencji*”. [13]

## 6. Realizacja

*"Wszystko powinno być tak proste, jak to tylko możliwe, ale nie prostsze." [43]*

Albert Einstein

W oparciu o założenia projektowe oraz koncepcje modelu wirtualnego świata gry i modelu Sztucznej Inteligencji, przystępujemy do realizacji pracy. W tworzonej przez nas grze „*Tank Wars*”<sup>133</sup> intelligentny i autonomiczny czołg porusza się po trójwymiarowym świecie, zaimplementowanym w środowisku programistycznym Microsoft XNA Game Studio. W pierwszym dziale przedstawimy proces przygotowania nowoczesnej grafiki komputerowej do realizacji podstawowych graficznych komponentów gry, takich jak: konstrukcja geometrii terenu gry; model czołgu, jego animacja i sterowanie; pozostałe modele występujące w grze; efekty wizualne oraz obsługa oddziaływań fizycznych obiektów. W drugim dziale przechodzimy do realizacji modelu Sztucznej Inteligencji, przedstawienia architektury intelligentnego agenta, maszyny stanów sterującej jego działaniami, opierającej się o rozmytym wnioskowaniu na podstawie zebranych obserwacji. W ostatnim dziale badamy miarę skuteczności działania intelligentnego agenta, poddając go symulacji w różnych stanach środowiska.

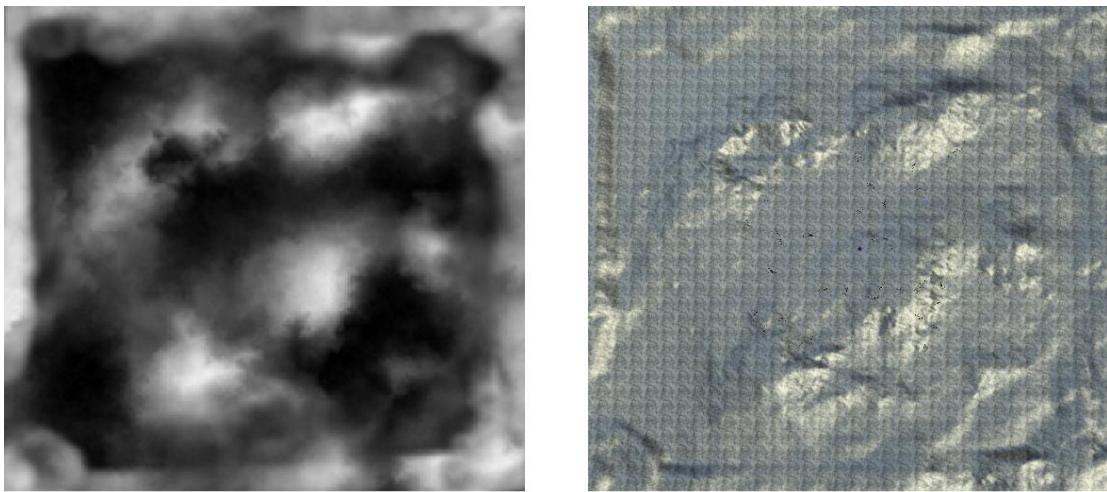
### 6.1 Nowoczesna grafika komputerowa w grze „*Tank Wars*”

- Przygotowanie wirtualnego świata gry

Pierwszym etapem prac jest wykonanie projektu terenu. Posłuży temu rozpatrzenie wymagań jakie stawia przed nami gra oraz analiza wstępnej koncepcji jej wyglądu. Projektując mapę terenu należy wziąć pod uwagę jej techniczne aspekty, jak skala, ukształtowanie terenu, czy ograniczenie możliwości wyjazdu poza nią. Na podstawie wstępnej koncepcji, przygotowana została bitmapa, wykonana w skali szarości charakteryzująca ukształtowanie obszaru. Wysokość wierzchołka tworzonego terenu określa kolor odpowiadającego mu piksela użytej bitmapy. Barwa czarno-szara reprezentuje tereny wklęsłe, położone niżej, natomiast jasno-biały odcień koloru szarego przedstawia obszary górzyste, wzniesienia. Ilustracja 6.1 przedstawia bitmapę i odpowiadający jej teren widoczny z lotu ptaka.

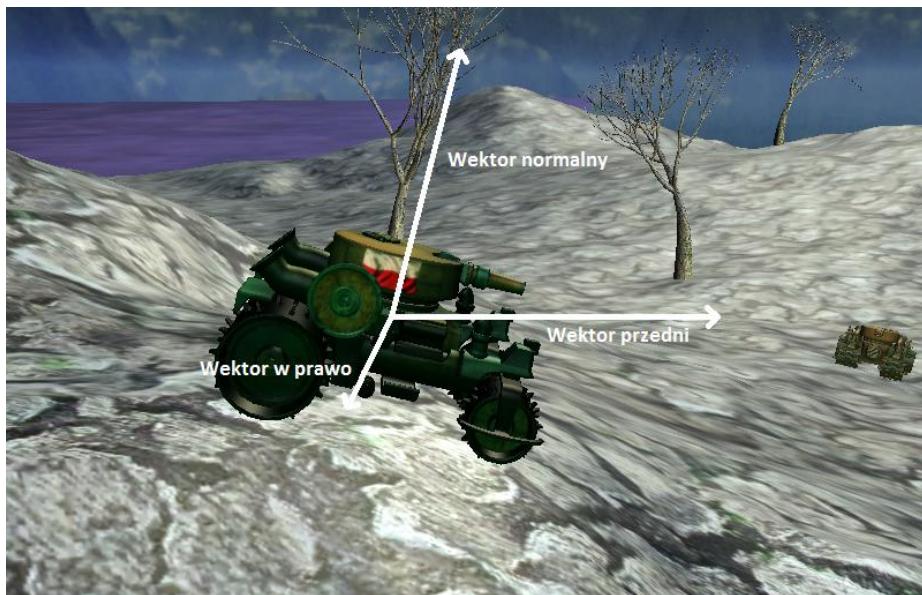
---

<sup>133</sup> Pochodzenie nazwy gry wyjaśnione jest we wprowadzeniu, w rozdziale 1.2 „*Cel pracy*”.



Ilustracja 6.1. Bitmapa i odpowiadający jej teren w grze „*Tank Wars*”.<sup>134</sup>

Miejsca styku koloru czarnego oraz szarego na bitmapie tworzą kontury wzniesień terenu. Ze względu na górzystą naturę terenu wektor normalny jednostek poruszających się po jego powierzchni będzie się zmieniał wraz z przemieszczaniem danej jednostki. Wektor ten zorientowany jest zgodnie z nachyleniem terenu, na którym znajduje się czołg. Wektor nachylenia poruszającej się jednostki otrzymywany jest z iloczynu wektorowego wektora normalnego oraz wektora prawnego<sup>135</sup>. Jest on prostopadły do określających go wektorów i wyznacza orientację czołgu. Umożliwia to otrzymanie bardziej realistycznego efektu dla obiektów poruszających się po wzgórzach, co przedstawia Ilustracja 6.2.



Ilustracja 6.2. Czołg na wzgórzu.

<sup>134</sup> Tekstura skał wykorzystana do pokrycia mapy pochodzi z tutoriala „*Collision with a Heightmap*” [46].

<sup>135</sup> Wektor prawy otrzymywany jest jako iloczyn wektorowy wektora normalnego oraz wektora przedniego.

Znając wektor normalny do powierzchni, wektor przedni, oraz bazując na założonej w koncepcji metodzie otrzymywania wysokości, możliwe jest zaimplementowanie naturalnego poruszania się czołgów po tak przygotowanej mapie.

Kolejnym etapem jest wypełnienie tła. Do zrealizowania tego kroku, zgodnie z założoną koncepcją, wykorzystany zostanie dużych rozmiarów sześciąan, którego ściany pokryte będą tekstonurą odległych gór i nieba. Skonstruowany teren, wraz z obiektyami po nim się poruszającymi i kamerą, będą znajdowały się w jego wnętrzu.

Niekonwencjonalna kamera zaprojektowania w poprzednim dziale, będzie śledzić główny czołg. Działającą na nią siłą sprężyny o zadanym współczynniku tłumienia oraz sprężystości, wpływ na jej sposób poruszania. Wszystkie obiekty na mapie będą rysowane zgodnie z jej perspektywistyczną projekcją, oraz będą przemieszczone z przestrzeni świata we współrzędne kamery zgodnie z wzorem (65) na transformacje modelu. Proces przycinania sceny oraz mapowania ekranu wykonywany jest automatycznie poprzez silnik graficzny XNA. Ilustracja 6.3 obrazuje działanie kamery.

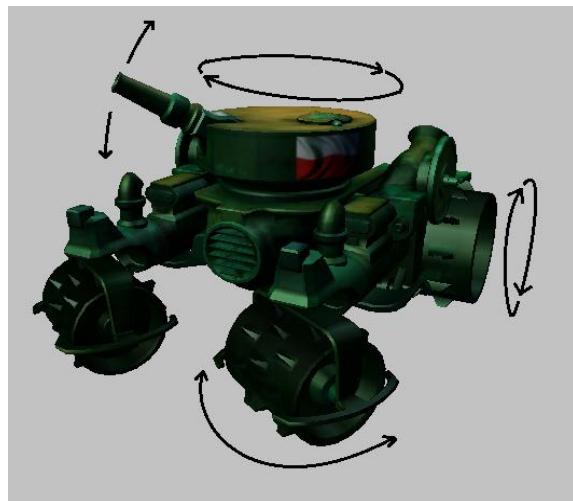


Ilustracja 6.3. Różne widoki z kamery pościgowej.

- Przygotowanie modeli

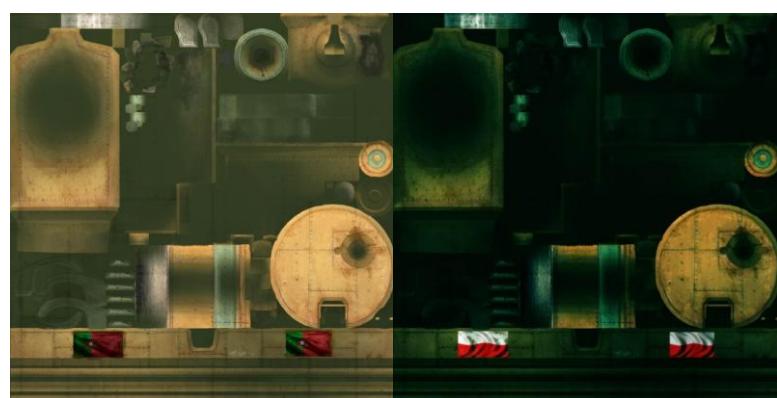
Wszystkie modele w grze reprezentowane są przez siatki trójkątów lub kwadratów. Każdy ich wierzchołek opisany jest przez zadany im kolor, który podlega zmianą wskutek procesów cieniowania. Na każdy obiekt w grze oddziaływa globalnie ustawione światło, oraz składowe światła rozproszonego, odbitego, oraz pochodzącego z otoczenia zgodnie ze wzorem (71) zastosowanego przez nas modelu oświetlenia Phonga. W celu nadania mrocznego charakteru, do gry wprowadzona została mgła. Każdy z modeli posiada indywidualnie ustawione parametry jej wyświetlanego. Tak przetworzone fragmenty obiektów po przejściu przez proces rasteryzacji, oraz szereg testów (patrz rozdział 4.2) stają się pikselami wyświetlonymi na ekranie.

Głównym obiektem geometrycznym w grze jest model czołgu<sup>136</sup>. Jest to kluczowa jednostka w grze. Dzięki zawartej w modelu hierarchicznej strukturze kości (patrz Ilustracja 5.5), możliwe jest zaimplementowanie animacji jego poszczególnych części. Przy przemieszczaniu się czołgu, jego koła obracają się, natomiast przy zmianie kierunku jazdy, ulegają skrętowi. Wieżyczka czołgu ulega obrotowi, wraz z którą obraca się jego działo. Działo, z którego wydawane są wystrzały może poruszać się w górę lub dół. Dodatkowo wieżyczce czołgu umieszczony jest włącz, który może ulegać otwarciu lub zamknięciu. Animowane elementy wraz z ich zakresem ruchu przedstawia Ilustracja 6.4.



Ilustracja 6.4. Model czołgu i jego animacja.<sup>137</sup>

Do modelu czołgu przygotowane zostały w edytorze graficznym dwie tekstury<sup>138</sup>. Główna jednostka posiada ciemno-zielony odcień z flagą polską po dwóch stronach wieżyczki. Wrogie jednostki posiadają pustynne barwy z flagą portugalską u boku. Tekstury czołgów przedstawia Ilustracja 6.5.



Ilustracja 6.5. Dwa rodzaje tekstury czołgu.<sup>136</sup>

<sup>136</sup> Geometria czołgu pochodzi z przykładu „Collision with a Heightmap” [46]

<sup>137</sup> Źródło własne.

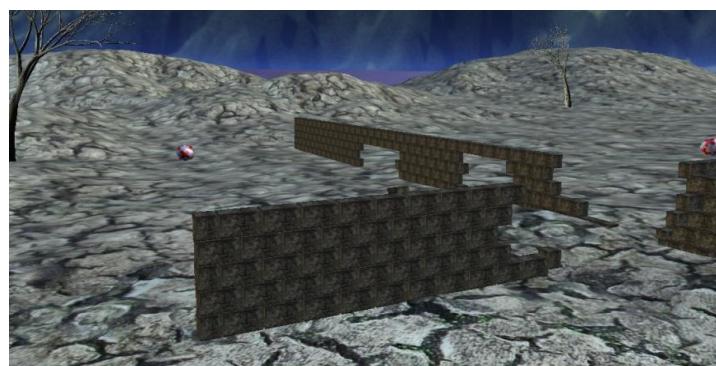
<sup>138</sup> Tekstury zostały opracowane na podstawie teksturow czołgu z przykładu „Collision with a Heightmap” [46]

Największym obiektem na terenie gry jest budynek kościoła. Znajduje się na środku planszy, a przed nim umieszczona jest statua orła<sup>139</sup> – główny celu do zniszczenia przez wrogie jednostki. Oprócz walorów wizualnych, budynek ma za zadanie utrudnić poruszanie się po obszarze inteligentnym czołgom. Ilustracja 6.6 prezentuje model kościoła.



Ilustracja 6.6. Model kościoła. [48]

W grze znajduje się kilka rodzajów obiektów wielokrotnie się powtarzających. Pierwszym z nich są kuliste apteczki, losowo pojawiające się na mapie. Kolejnym, bardziej skomplikowany jest model ściany. Zgrupowane obiekty, tworzące jej strukturę, współdzielą własności fizyczne, oraz wizualne. Na każdą ze ścian składa się kilka, do kilkudziesięciu mały prostopadłościennych cegieł. Ze względu na ogromną ich całkowitą liczbę na całej mapie, w celu zwiększenia wydajności gry, do ich rysowania wykorzystywany jest jeden, współdzielony model prostopadłościennej cegły pokrytej tekstrurą. Technika ta, zwana geometrią chwilową, opisana i zaprezentowana jest w rozdziale 4.2. Ilustracja 6.7 prezentuje fragmenty zniszczonych murów.



Ilustracja 6.7. Konstrukcja modelu ściany.<sup>140</sup>

Drugim rodzajem, powtarzających się modeli są wszechobecne drzewa. Do ich generacji wykorzystana została specjalnie przygotowana biblioteka dla oprogramowania XNA.

---

<sup>139</sup> Model geometryczny orła pochodzi z *Artist-3D*. [52]

<sup>140</sup> Źródło własne.

Charakteryzuje się możliwością tworzenia tego samego typu drzew, różniących się ilością, czy rozpiętością gałęzi. Ich gęstość, rozłożenie jak i miejsce występowania są parametryzowane. Aby pasowały do mrocznego charakteru sceny, mgły i skalistego terenu, zostały zaimplementowane bez liści. Ilustracja 6.8 przedstawia przykład wygenerowanych drzew.



Ilustracja 6.8. Automatycznie generowane różne modele drzew. [56]

- Przygotowanie silnika cząsteczek

Bardzo istotnym elementem, nadającym jakości gry i cieszącym oko gracza, są efekty specjalne. W znacznym stopniu zapewniają nam je symulacje cząsteczek. Ilustracja 6.9 obrazuje przykład ich wykorzystania – dwa czołgi walczące na polu bitwy.

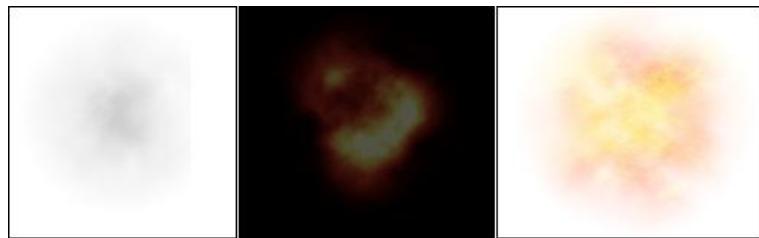


Ilustracja 6.9. Symulacja cząsteczek przy wystrzale i trafieniu.<sup>141</sup>

Zaimplementowany system cząsteczek (schemat Ilustracja 5.6) umożliwia symulację wystrzału ognia, wybuchów, dymu, czy kurzu. Na każdy taki efekt składają się setki tysięcy małych cząsteczek reprezentowanych przez bitmapy. Ilustracja 6.10 przedstawia trzy główne cząsteczki, składające się na większość efektów związanych z wybuchami.

---

<sup>141</sup> Źródło własne.



Ilustracja 6.10. Bitmapy cząsteczek dymu, wybuchu i ognia.<sup>142</sup>

Ilustracja 6.11 prezentuje działanie emitera cząstek. Wraz z wystrzałem, strumień cząstek podąża za wystrzelonym pociskiem, zostawiając po sobie smugę dymu. Pocisk po uderzeniu w drugi obiekt, ścianę, skały, czy budynek ulega eksplozji. Wybuchowi towarzyszą płomienie ognia oraz kłęby dymu.



Ilustracja 6.11. Działanie emitera cząsteczek przy wystrzale pocisku.<sup>143</sup>

Kolejnym przykładem zastosowanym w grze, jest symulacją cząsteczek kurzu wydobywających się z przyspieszającego pojazdu. Ilustracja 6.12 przedstawia obłoki dymu tworzonych przez koła jadącego czołgu.



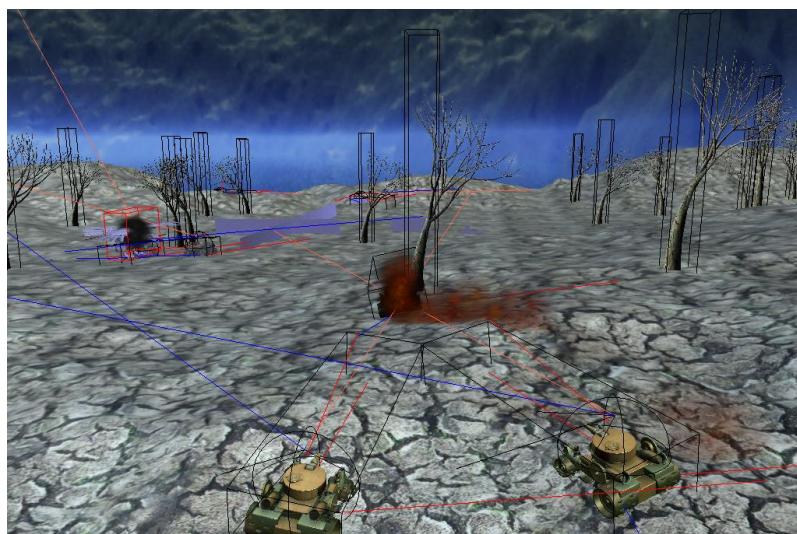
Ilustracja 6.12. Symulacją cząsteczek kurzu.<sup>142</sup>

<sup>142</sup> Bitmapy cząsteczek pochodzą z przykładu „Particle3D” [57]

<sup>143</sup> Źródło własne.

- Przygotowanie fizyki gry

Trójwymiarowym światem steruje sztuczna inteligencja w oparciu o fizykę gry. System percepji, pomiaru odległości, unikania kolizji bazuje na informacjach podawanych przez bryły brzegowe, sferyczne obszary postrzegania, czy wektory kierunku. Fizyka została również zastosowana do obliczania toru lotu pocisku, wykrywania jego zderzeń z obcymi jednostkami, czy obiektami, obsługi kolizji czołgów i ich poruszania się. Wszystkie obiekty, zarówno statyczne jak i dynamiczne, posiadają nadane im bryły brzegowe. Ilustracja 6.13 prezentuje ukryty świat fizyki.



Ilustracja 6.13. Bryły brzegowe otaczające obiekty w grze.<sup>144</sup>

Losowo pojawiająca się na mapie gry apteczka jest dynamicznym obiektem fizycznym o zadanej masie i właściwościach materiału z jakiego jest wykonana. Upada ona na pokryty statyczną siatką teren gry, uniemożliwiający jej dalszy spadek. Ze względu na swój kulisty charakter i niski współczynnik tarcia toczy się ona po górzystym terenie gry pod wpływem siły grawitacji, lub kolizji z innymi obiektami.



Ilustracja 6.14. Tocząca się po terenie kulista apteczka.<sup>143</sup>

Efektowne działanie fizyki demonstrują zderzenia czołgów z ceglastymi murami. Przejedżający przez nie czołg o dużo większej masie kruszy ścianę, rozrzucając dużo

---

<sup>144</sup> Źródło własne.

lżejsze od niego cegły po pobliskim terenie. Ilustracja 6.15 przedstawia scenę przejazdu czołgów przez mur.



Ilustracja 6.15. Kolizja czołgu ze ścianą.<sup>145</sup>

Wystrzelony przez czołg pocisk, o zadanej prędkości początkowej, masie, oraz fizycznych właściwościach materiału, porusza się w przestrzeni po krzywej balistycznej. W przypadku trafienia przez pocisk muru, ulega on wybuchowi. Siła uderzenia, spotęgowana przez zaimplementowaną eksplozję, w efektowny sposób niszczy napotkane przez pocisk mury. Ilustracja 6.16 przedstawia przykład takiej eksplozji.



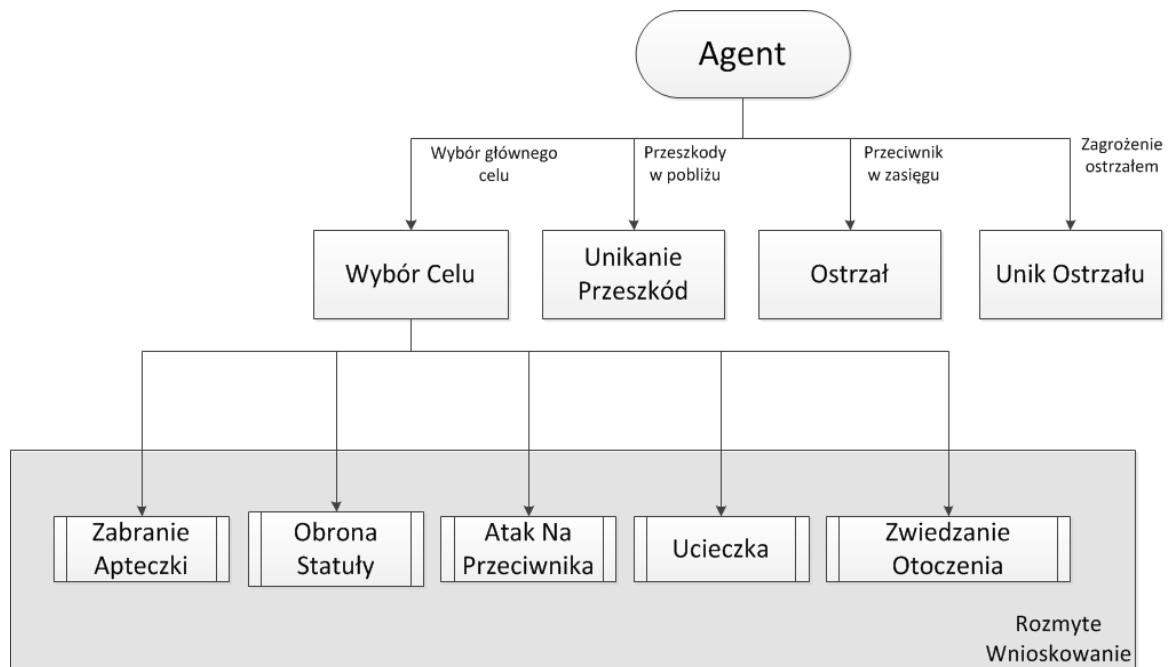
Ilustracja 6.16. Eksplozja pocisku przy zderzeniu ze ścianą.<sup>144</sup>

---

<sup>145</sup> Źródło własne.

## 6.2 Model Sztucznej Inteligencji

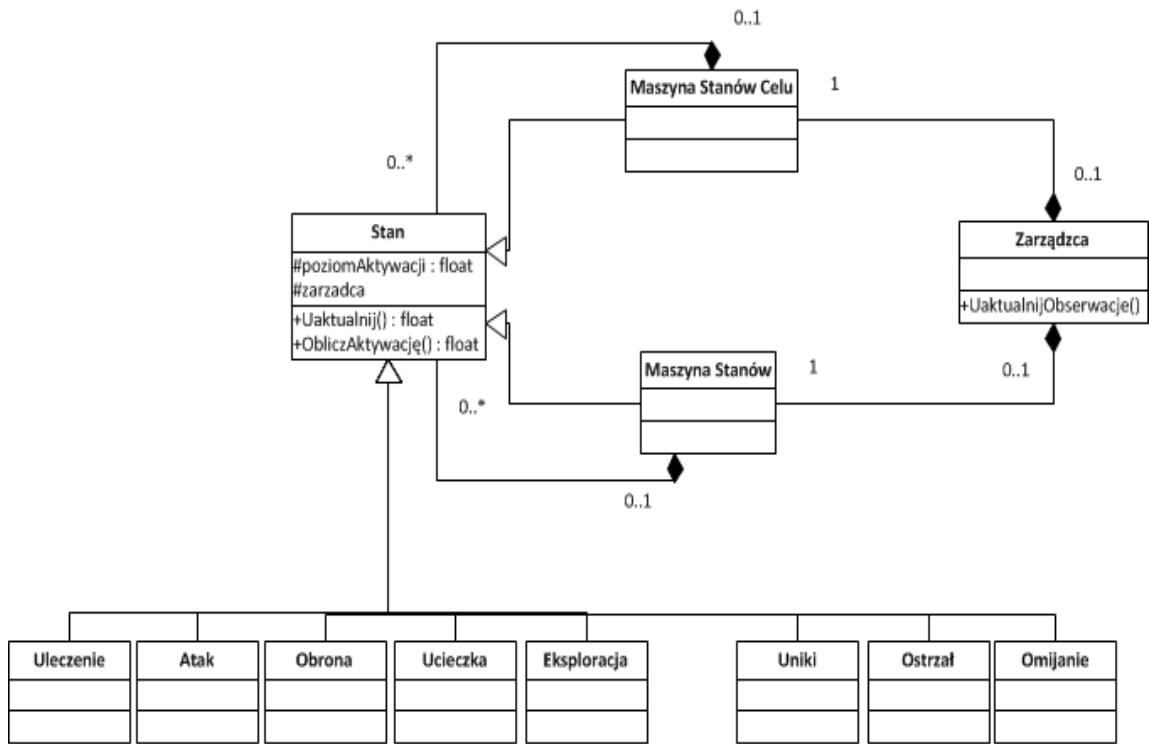
Architektura Agenta opiera się w głównej mierze o pewnego rodzaju hierarchiczną maszynę stanów. Stworzone zostały dwie maszyny działające na trochę różnych zasadach, jedna maszyna jest również jednym ze stanów drugiej, co jest przedstawione na Ilustracja 6.17. Maszyna stanów "Wybór Celu" ma za zadanie kierować głównymi pragnieniami agenta, poszczególne stany wyliczają swój poziom aktywacji na podstawie zebranych obserwacji, następnie tylko jeden stan o najwyższym poziomie aktywacji może zostać uruchomiony. Jest tutaj pewne podobieństwo do drzewa zachowań z gry „Halo”, ponieważ stany muszą konkurować o to, który ma zadziałać. Jednak w tym przypadku rozdzielenie priorytetu i aktywacji zostało zamienione na jeden parametr czyli poziom aktywacji. Nadrzędna maszyna stanów zawiera 4 stany, które są od siebie nie zależne, w tym przypadku nie ma konkurencji, wszystkie mogą działać w tym samym czasie. Działanie tej maszyny można interpretować jako krótkoterminowe, nagłe pragnienia agenta, takie jak omijanie nie spodziewanych przeszkód, unikanie nagłego strzału oraz ostrzeliwanie napotkanego przeciwnika.



Ilustracja 6.17. Schemat Maszyny Stanów sterującej robotem.<sup>146</sup>

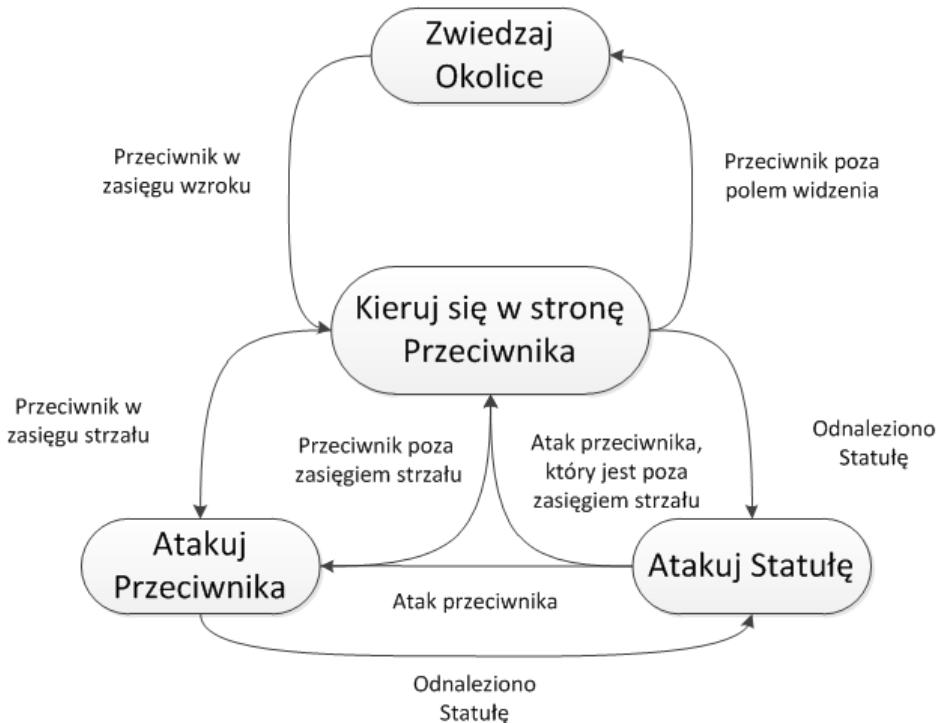
Ilustracja 6.18 przedstawia sposób implementacji powyższej maszyny stanów w postaci diagramu UML. Jak można zauważyć, stany w obu typach maszyn nie różnią się od siebie, to sama maszyna stanów decyduje o tym jak je wykorzystać. Każdy stan w metodzie "Uaktualnij" wylicza odpowiadający mu wektor siły kierujący robotem. Następnie wszystkie działające stany sumują wyliczone wektory, nadając mu odpowiedni kierunek.

<sup>146</sup> Źródło własne.



Ilustracja 6.18. Diagram UML Maszyny Stanów sterującej robotem.<sup>147</sup>

Agent o zaprezentowanej powyżej architekturze musi zmierzyć się z przeciwnikami, kontrolowanymi przez bardzo prostą maszynę stanów, którą przedstawia Ilustracja 6.19.



Ilustracja 6.19. Schemat Maszyny Stanów sterującej przeciwnikami robota.<sup>148</sup>

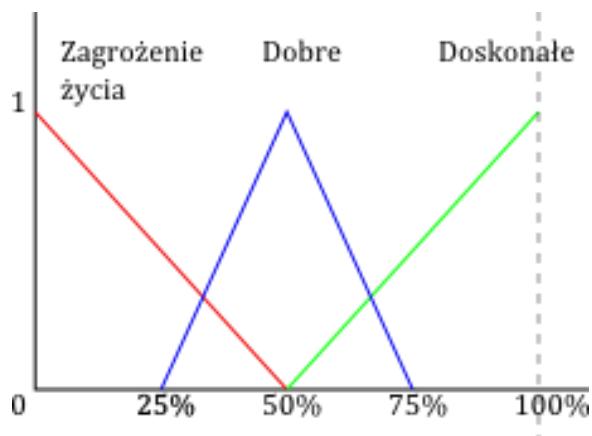
<sup>147</sup> Źródło własne.

- Wnioskowanie na podstawie zebranych obserwacji

Atak na przeciwnika oraz Ucieczka - poziom aktywacji tych stanów jest określany na podstawie poziomu zdrowia agenta oraz napotkanego przeciwnika. Poziom zdrowia (Ilustracja 6.20) jest informacją nie precyzyjną, dlatego do wyliczenia poziomu agresji (Ilustracja 6.21) w stosunku do przeciwnika wykorzystane zostało rozmyte wnioskowanie. Tabela 5 przedstawia zbiór 9 reguł na podstawie, który podejmowana jest decyzja. Ponadto gdy agent będzie miał odpowiednio niski poziom agresji, będzie próbował uciekać, wyjątkiem jest sytuacja gdzie zagrożona jest statua i nie wolno mu tego zrobić.

Tabela 5 Zbiór reguł sterujących agresją robota.<sup>149</sup>

Nr. Reguły	Własne zdrowie	Zdrowie przeciwnika	Poziom agresji
1	Doskonałe	Doskonałe	Zwyczajny
2	Doskonałe	Dobre	Wysoki
3	Doskonałe	Zagrożenie życia	Wysoki
4	Dobre	Doskonałe	Niski
5	Dobre	Dobre	Zwyczajny
6	Dobre	Zagrożenie życia	Wysoki
7	Zagrożenie życia	Doskonałe	Niski
8	Zagrożenie życia	Dobre	Niski
9	Zagrożenie życia	Zagrożenie życia	Zwyczajny



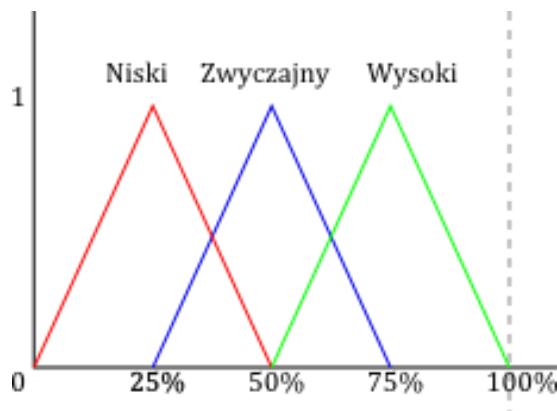
Ilustracja 6.20. Graficzne przedstawienie funkcji reprezentujących zdrowie robota.<sup>150</sup>

Do wyliczenia wektorów siły posłużyły odpowiednio algorytmy zachowań sterujących pościgu oraz ucieczki.

<sup>148</sup> Źródło własne.

<sup>149</sup> Źródło własne.

<sup>150</sup> Źródło własne.

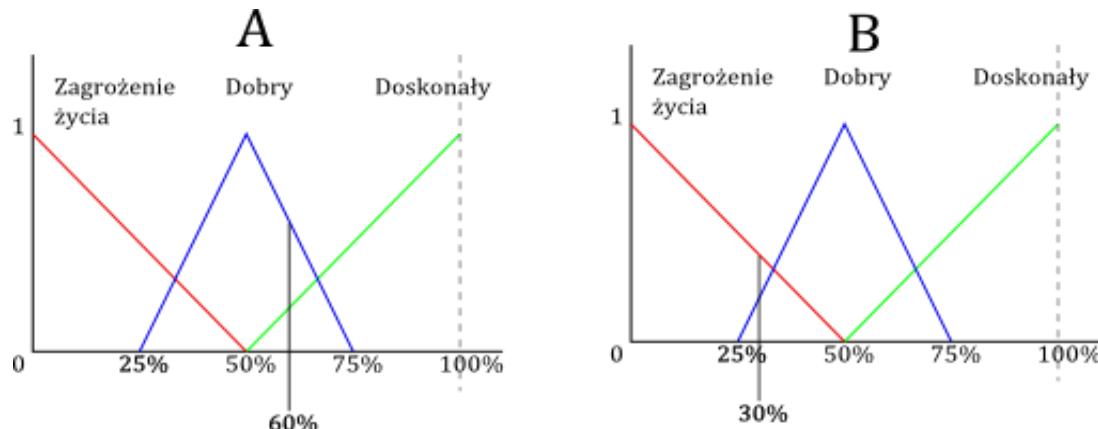


Ilustracja 6.21. Graficzne przedstawienie funkcji reprezentujących poziom agresji.<sup>151</sup>

Przykład 1:

Dane wejściowe:

$\bar{x}_1$  – Własny stan zdrowia: 60% (0,6);  $\bar{x}_2$  – Stan zdrowia przeciwnika 30% (0,3).



Ilustracja 6.22. Odwzorowanie rozmytego sygnału wejściowego na funkcje zdrowia A) Robota i B) jego przeciwnika.<sup>152</sup>

Sygnal wejściowy jest rozmywany metodą typu singleton na podstawie wzoru (86). Jak pokazuje Ilustracja 6.22 sygnał wejściowy przecina dwa zbiory rozmyte definiujące zdrowie robota ("Dobry", "Doskonały") oraz dwa zbiory rozmyte definiujące zdrowie jego przeciwnika ("Zagrożenie życia", "Dobry") tzn. że tylko te zbiory będą miały wpływ na wynik. Zgodnie z tabelą reguł (Tabela 5) te, które zadziałają to: 2, 3, 5, 6. Na podstawie wzoru (14) można wyliczyć punkty przecięcia:

$$\mu_{A^{Dobry}}(0,6) = \frac{0,75 - 0,6}{0,75 - 0,5} = \frac{0,15}{0,25} = 0,6,$$

<sup>151</sup> Źródło własne.

<sup>152</sup> Źródło własne.

$$\mu_{A^{\text{Dokona} \text{ by}}} (0,6) = \frac{0,6 - 0,5}{1 - 0,5} = \frac{0,1}{0,5} = 0,2,$$

$$\mu_{B^{\text{Zagro} \text{ \v{z}enie}}} (0,3) = \frac{0,5 - 0,3}{0,5 - 0} = \frac{0,2}{0,5} = 0,4,$$

$$\mu_{B^{\text{Dobry}}} (0,3) = \frac{0,3 - 0,25}{0,5 - 0,25} = \frac{0,05}{0,25} = 0,2.$$

Wnioskowanie zostanie przeprowadzone zgodnie ze wzorem (103). Na podstawie wzoru (104) otrzymujemy z poszczególnych reguł:

$$\text{Regu\l{}a 2: } \mu_{A_1^2 \times A_2^2} (\bar{x}_1, \bar{x}_2) = \min(0,2; 0,2) = 0,2,$$

$$\text{Regu\l{}a 3: } \mu_{A_1^3 \times A_2^3} (\bar{x}_1, \bar{x}_2) = \min(0,2; 0,4) = 0,2,$$

$$\text{Regu\l{}a 5: } \mu_{A_1^5 \times A_2^5} (\bar{x}_1, \bar{x}_2) = \min(0,6; 0,2) = 0,2,$$

$$\text{Regu\l{}a 6: } \mu_{A_1^6 \times A_2^6} (\bar{x}_1, \bar{x}_2) = \min(0,6; 0,4) = 0,4.$$

Ponadto:

$$\mu_{B^2}(y) = \mu_{B^3}(y) = \mu_{B^6}(y) = \begin{cases} \frac{y - 0,5}{0,75 - 0,5}, & \text{dla } 0,5 < y \leq 0,75, \\ \frac{1 - y}{1 - 0,75}, & \text{dla } 0,75 < y \leq 1. \end{cases}$$

$$\mu_{B^5}(y) = \begin{cases} \frac{y - 0,25}{0,5 - 0,25}, & \text{dla } 0,25 < y \leq 0,5, \\ \frac{0,75 - y}{0,75 - 0,5}, & \text{dla } 0,5 < y \leq 0,75. \end{cases}$$

Rezultatem poszczególnych reguł będą zbiory określone na podstawie wzoru (105):

$$\text{Regu\l{}a 2: } \mu_{\bar{B}^2}(y) = \min(0,2, \mu_{B^2}(y)),$$

$$\text{Regu\l{}a 3: } \mu_{\bar{B}^3}(y) = \min(0,2, \mu_{B^3}(y)),$$

$$\text{Regu\l{}a 5: } \mu_{\bar{B}^5}(y) = \min(0,2, \mu_{B^5}(y)),$$

$$\text{Regu\l{}a 6: } \mu_{\bar{B}^6}(y) = \min(0,4, \mu_{B^6}(y)).$$

Następnie na podstawie wzoru (106) otrzymujemy zbiór wynikowy, który jest sumą zbiorów otrzymanych z każdej z 4 reguł:

$$\mu_{B'}(y) = \max(\mu_{\bar{B}^2}(y); \mu_{\bar{B}^3}(y); \mu_{\bar{B}^5}(y); \mu_{\bar{B}^6}(y)).$$

Na podstawie wzoru (113) zbiór  $\mu_{B'}(y)$  zostanie wyostrzony, przez co otrzymamy konkretną wartość  $\bar{y}$ . Do wyliczenia przybliżonych wartości potrzebnych do wzoru całek posłuży metoda trapezów, która polega na podzieleniu obszaru całkowania na  $n+1$  punktów, znajdujących się od siebie w odległości  $dy$ , dla każdego punktu  $y_i$ , gdzie  $i \in (0, \dots, n)$ , zostanie wyliczona odpowiadająca mu wartość  $f_i$  dla każdej z całek.

Sąsiednie punkty oraz ich wartości tworzą trapez, którego pole można obliczyć według wzoru:

$$P_i = \frac{f_{i-1} + f_i}{2} dy, \text{ gdzie } i \in (1, \dots, n). \quad (116)$$

Suma wszystkich pól jest poszukiwanym przybliżeniem całki. Dla przykładu zostanie przyjęte  $dy = 0,05$ ,  $y_0 = 0,25$  i  $y_n = 1$ . Ponadto:

$$f_i^1 = \mu_{B'}(y_i),$$

$$f_i^2 = y_i \mu_{B'}(y_i).$$

Na podstawie wzoru (116):

$$P_i^1 = \frac{f_{i-1}^1 + f_i^1}{2} dy,$$

$$P_i^2 = \frac{f_{i-1}^2 + f_i^2}{2} dy.$$

Tabela 6 Zestawienie wartości poszczególnych funkcji oraz pól trapezów.<sup>153</sup>

i	0	1	2	3	4	5	6	7
$y_i$	0,25	0,3	0,35	0,4	0,45	0,5	0,55	0,6
$\mu_{B'}(y_i)$	0	0,2	0,2	0,2	0,2	0,2	0,2	0,4
$P_i^1$		0,005	0,01	0,01	0,01	0,01	0,01	0,015
$y_i \mu_{B'}(y_i)$	0	0,06	0,07	0,08	0,09	0,1	0,11	0,24
$P_i^2$		0,0015	0,00325	0,00375	0,00425	0,00475	0,00525	0,00875
i	8	9	10	11	12	13	14	15
$y_i$	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1
$\mu_{B'}(y_i)$	0,4	0,4	0,4	0,4	0,4	0,4	0,2	0
$P_i^1$	0,02	0,02	0,02	0,02	0,02	0,02	0,015	0,005
$y_i \mu_{B'}(y_i)$	0,26	0,28	0,3	0,32	0,34	0,36	0,19	0
$P_i^2$	0,0125	0,0135	0,0145	0,0155	0,0165	0,0175	0,01375	0,00475

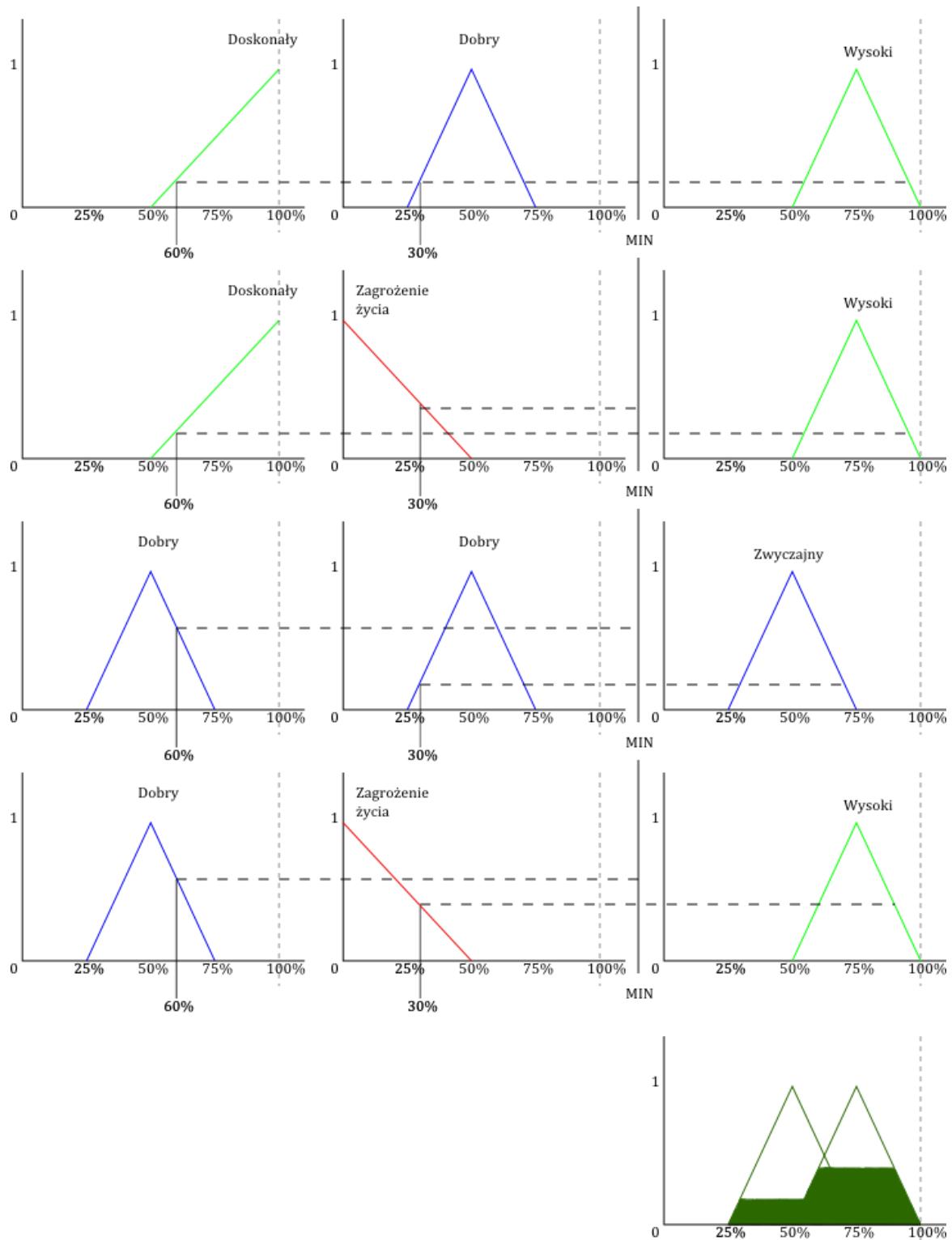
$$\bar{y} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy} = \frac{\sum_{i=1}^n P_i^2}{\sum_{i=1}^n P_i^1} = \frac{0,14}{0,21} = 0, (66).$$

Oczywiście nie jest to zbyt dokładny wynik, ponieważ dla przykładu użyty został bardzo duży krok  $dy$ . Poziom agresji jest wyższy niż 0,5 co oznacza pościg za przeciwnikiem, wartości poniżej 0,5 oznaczają ucieczkę. Wynik należy jeszcze przeskalać na zakres o wielkości 1:

$$\bar{y}_{Atak} = \frac{0,66 - 0,5}{0,5} = 0,32.$$

<sup>153</sup> Źródło własne.

Poziom aktywacji stanu ataku wynosi 0,32. Ilustracja 6.23 prezentuje graficzną interpretację powyższego wnioskowania.



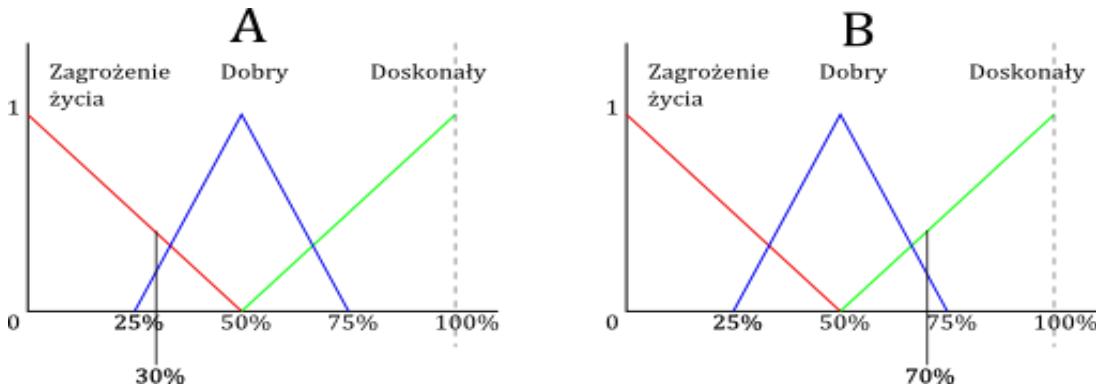
Ilustracja 6.23. Graficzna prezentacja przeprowadzonego wnioskowania przykładowu 1.<sup>154</sup>

<sup>154</sup> Źródło własne.

Przykład 2:

Dane wejściowe:

$\bar{x}_1$  – Własny stan zdrowia: 30% (0,3);  $\bar{x}_2$  – Stan zdrowia przeciwnika 70% (0,7).



Ilustracja 6.24. Odwzorowanie rozmytego sygnału wejściowego na funkcje zdrowia A) Robota i B) jego przeciwnika.<sup>155</sup>

Analogicznie jak w poprzednim przykładzie na podany na wejście sygnał zareagują 4 reguły, tym razem będą to reguły o numerach 4,5,7 i 8. Wyznaczamy punkty przecięcia sygnału wejściowego oraz odpowiednich zbiorów rozmytych:

$$\mu_{A^{Dobry}}(0,3) = \frac{0,3 - 0,25}{0,5 - 0,25} = \frac{0,05}{0,25} = 0,2,$$

$$\mu_{A^{Zagrożenie}}(0,3) = \frac{0,5 - 0,3}{0,5 - 0} = \frac{0,2}{0,5} = 0,4,$$

$$\mu_{B^{Dobry}}(0,7) = \frac{0,75 - 0,7}{0,75 - 0,5} = \frac{0,05}{0,25} = 0,2,$$

$$\mu_{B^{Doskonały}}(0,7) = \frac{0,7 - 0,5}{1 - 0,5} = \frac{0,2}{0,5} = 0,4.$$

Następnie wyznaczamy  $\mu_{A_1^k \times A_2^k}$  dla  $k = 4,5,7,8$ :

$$\text{Reguła 4: } \mu_{A_1^4 \times A_2^4}(\bar{x}_1, \bar{x}_2) = \min(0,2; 0,4) = 0,2,$$

$$\text{Reguła 5: } \mu_{A_1^5 \times A_2^5}(\bar{x}_1, \bar{x}_2) = \min(0,2; 0,2) = 0,2,$$

$$\text{Reguła 7: } \mu_{A_1^7 \times A_2^7}(\bar{x}_1, \bar{x}_2) = \min(0,4; 0,4) = 0,4,$$

$$\text{Reguła 8: } \mu_{A_1^8 \times A_2^8}(\bar{x}_1, \bar{x}_2) = \min(0,4; 0,2) = 0,2.$$

Ponadto:

$$\mu_{B^4}(y) = \mu_{B^7}(y) = \mu_{B^8}(y) = \begin{cases} \frac{y - 0}{0,25 - 0}, & \text{dla } 0 < y \leq 0,25, \\ \frac{0,5 - y}{0,5 - 0,25}, & \text{dla } 0,25 < y \leq 0,5. \end{cases}$$

<sup>155</sup> Źródło własne.

$$\mu_{B^5}(y) = \begin{cases} \frac{y - 0,25}{0,5 - 0,25}, & \text{dla } 0,25 < y \leq 0,5, \\ \frac{0,75 - y}{0,75 - 0,5}, & \text{dla } 0,5 < y \leq 0,75. \end{cases}$$

W rezultacie wyniki reguł przyjmują postać:

$$\text{Reguła 4: } \mu_{\bar{B}^4}(y) = \min(0,2, \mu_{B^4}(y)),$$

$$\text{Reguła 5: } \mu_{\bar{B}^5}(y) = \min(0,2; \mu_{B^5}(y)),$$

$$\text{Reguła 7: } \mu_{\bar{B}^7}(y) = \min(0,4; \mu_{B^7}(y)),$$

$$\text{Reguła 8: } \mu_{\bar{B}^8}(y) = \min(0,2; \mu_{B^8}(y)).$$

Następnie wyliczony zostaje zbiór wynikowy:

$$\mu_{B'}(y) = \max(\mu_{\bar{B}^4}(y); \mu_{\bar{B}^5}(y); \mu_{\bar{B}^7}(y); \mu_{\bar{B}^8}(y)).$$

Kolejnym krokiem jak w poprzednim przykładzie będzie wyliczenie całek potrzebnych do wyostrzenia zbioru  $\mu_{B'}(y)$  przyjęte zostanie  $dy = 0,05$ ,  $y_0 = 0$  oraz  $y_n = 0,75$ .

Tabela 7 Zestawienie wartości poszczególnych funkcji oraz pól trapezów.<sup>156</sup>

i	0	1	2	3	4	5	6	7
$y_i$	0	0,05	0,1	0,15	0,2	0,25	0,3	0,35
$\mu_{B'}(y_i)$	0	0,2	0,4	0,4	0,4	0,4	0,4	0,4
$P_i^1$		0,005	0,015	0,02	0,02	0,02	0,02	0,02
$y_i \mu_{B'}(y_i)$	0	0,01	0,04	0,06	0,08	0,1	0,12	0,14
$P_i^2$		0,00025	0,00125	0,0025	0,0035	0,0045	0,0055	0,0065
i	8	9	10	11	12	13	14	15
$y_i$	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75
$\mu_{B'}(y_i)$	0,4	0,2	0,2	0,2	0,2	0,2	0,2	0
$P_i^1$	0,02	0,015	0,01	0,01	0,01	0,01	0,01	0,005
$y_i \mu_{B'}(y_i)$	0,16	0,09	0,1	0,11	0,12	0,13	0,14	0
$P_i^2$	0,0075	0,00625	0,00475	0,00525	0,00575	0,00625	0,00675	0,0035

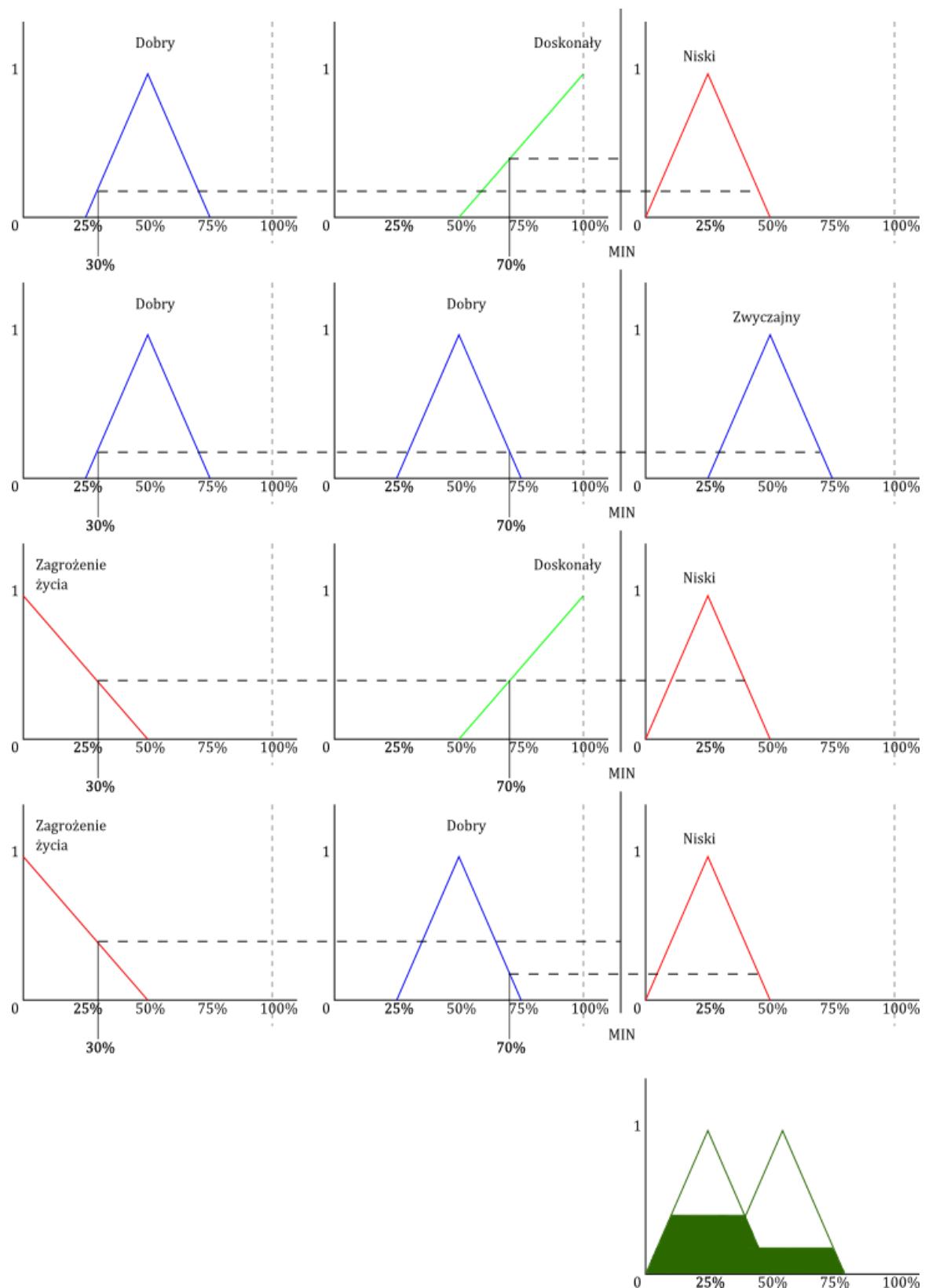
$$\bar{y} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy} = \frac{\sum_{i=1}^n P_i^2}{\sum_{i=1}^n P_i^1} = \frac{0,07}{0,21} = 0, (33).$$

Tym razem  $\bar{y}$  jest poniżej 0,5 to oznacza ucieczkę przed niebezpieczeństwem:

$$\bar{y}_{Ucieczka} = \frac{0,5 - 0,33}{0,5} = 0,34.$$

Poziom aktywacji stanu ucieczki wynosi 0,34. Ilustracja 6.25 prezentuje graficzną interpretację powyższego przykładu.

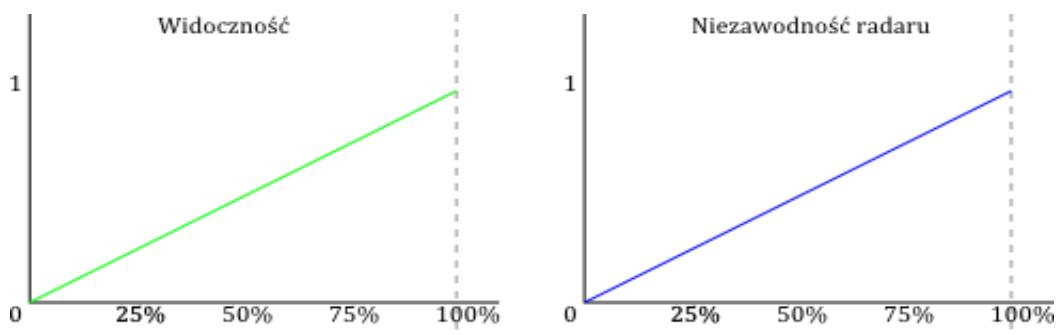
<sup>156</sup> Źródło własne.



Ilustracja 6.25. Graficzna prezentacja przeprowadzonego wnioskowania przykładowu 2.<sup>157</sup>

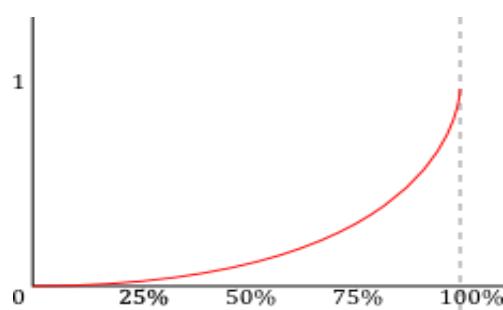
<sup>157</sup> Źródło własne.

Apteczka oraz statua - oba stany są do siebie bardzo podobne jeden odpowiada za zbieranie apteczek, natomiast drugi za sprawdzanie czy statua nie jest zagrożona. To zadanie również można rozpatrywać w ramach rozmytego wnioskowania, każde zadanie reprezentuje jedna reguła. Dla apteczki "Jeżeli odniosłem dużo zniszczeń I informacja z radaru jest niezawodna To jadę po apteczkę", gdzie wiarygodność radaru rozumiemy przez odległość apteczki od agenta, maksymalny zasięg radaru wynosi R. Przy maksymalnej odległości niezawodność jest na poziomie 0%, natomiast gdy apteczka znajduje się przy robocie niezawodność jest równa 100%. Dla statuy reguła brzmi "Jeżeli statua odniosła duże obrażenia I informacja wizualna jest wiarygodna To broń statuy". Tak jak w poprzednim przypadku im dalej agent znajduje się od statuy tym bardziej jego wzrok może zawodzić i tym mniejsza będzie pewność co do obrażeń odniesionych przez statuę. Ilustracja 6.26 prezentuje funkcje reprezentujące widoczność statuy oraz niezawodność radaru co do apteczki.



Ilustracja 6.26. Funkcje mapujące odległość apteczki oraz statuy od robota.<sup>158</sup>

Zasada działania jest zbliżona do funkcji wagi w grze „The Sims”. „Zasięg radaru” oraz „Zasięg wzroku” są funkcjami wagowymi, które mapują odległość robota od statuy oraz apteczki na pewien współczynnik z zakresu (0,1). Natomiast funkcja wagowa zniszczeń Ilustracja 6.27 określa procentowy poziom zniszczeń robota lub statuy. Podobnie jak w grze „The Sims” na podstawie tych funkcji możemy wyliczyć jak bardzo robot potrzebuje się uleczyć lub bronić statuy. Do zaprezentowania zasady działania, posłużą poniższe przykłady.



Ilustracja 6.27. Funkcja  $(x)^2$  reprezentująca zniszczenia robota oraz statuy.<sup>157</sup>

<sup>158</sup> Źródło własne.

Przykład dla apteczki:

Widoczność wynosi 50% ( $x_1 = 0,5$ ), natomiast zniszczenia wynoszą 75% ( $x_2 = 0,75$ ). Na podstawie funkcji widoczności oraz zniszczeń, zostaną wyliczone wagi odpowiadające podanym wartością. Dla  $x_1$  na podstawie wzoru (13) otrzymamy  $\mu_1 = 0,5$ , a dla  $x_2$

$$\mu_2 = \left( \frac{x_2 - a}{b - a} \right)^2 = (0,75)^2 = 0,5625.$$

Następnie według wzoru (64), można wyliczyć poziom aktywacji:

$$y = \frac{\sum_{i=1}^2 \mu_i x_i}{\sum_{i=1}^2 \mu_i} = \frac{0,5 * 0,5 + 0,75 * 0,5625}{0,5 + 0,5625} \approx 0,63.$$

Przykład dla statuy:

Niezawodność radaru można przyjąć jako 50% ( $x_1 = 0,5$ ) oraz zniszczenia statuy na poziomie 25% ( $x_2 = 0,25$ ), odpowiadające danym wartością funkcje wagi są następujące:

$$\mu_1 = 0,5$$

$$\mu_2 = \left( \frac{x_2 - a}{b - a} \right)^2 = (0,25)^2 = 0,0625.$$

Następnie według wzoru (64), wyliczony zostanie poziom aktywacji stanu:

$$y = \frac{\sum_{i=1}^2 \mu_i x_i}{\sum_{i=1}^2 \mu_i} = \frac{0,5 * 0,5 + 0,25 * 0,0625}{0,5 + 0,0625} \approx 0,47.$$

Przy podanych powyżej danych wybrany został by stan zebrania apteczki, co przy przyjrzeniu się odpowiednim wartościom wejściowym jest uzasadnione. Do wyliczenia wektora siły przy obu stanach posłużył algorytm podążania.

**Zwiedzanie otoczenia** - Może się zdarzyć, że żaden ze stanów zaprezentowanych powyżej nie będzie aktywny. W takim przypadku uruchamiany jest stan "Zwiedzanie otoczenia", którego zadaniem jest wcześniejsze zlokalizowanie przeciwników. Robot porusza się w przypadkowym kierunku, jednak nie może stracić statuy z zasięgu wzroku. Do wyliczenia odpowiedniego wektora siły posłużyło zachowanie sterujące o nazwie "Błędzenie".

Unikanie ostrzału, ostrzał, oraz unikanie przeszkód to proste stany, które wyliczają odpowiednie wektory siły przy zaistniałych warunkach w otoczeniu. I tak odpowiednio dla unikania ostrzału zastosowano zachowanie sterujące unikania. Do unikania przeszkód wykorzystano dwa typy algorytmów, jeden do omijania nie wielkich przeszkód, takich jak drzewa czy inne czołgi w grze oraz drugi do omijania większych przeszkód takich jak kościół, czy mury. Stan ostrzału ma za zadanie tylko nakierować ostrzał na przeciwnika jeżeli ten znajdzie się w zasięgu strzału.

- Mechanizm wykonawczy

Każdy ze stanów wylicza odpowiednią funkcję aktywacji z zakresu od 0 do 1. Jeżeli stan zostanie wybrany przez maszynę do działania, zostaje wywołana funkcja aktualizacji, która wylicza odpowiedni wektor siły, jaka powinna działać na agenta by dążył do swojego celu tak jak to przedstawiono w koncepcji sterowania. Następnie wszystkie wynikowe siły są ze sobą sumowane oraz przekładane bezpośrednio na ruch. Wyjątkiem jest tutaj tylko stan "Ostrzał", który odpowiada za poruszanie wieżyczką i strzelanie do przeciwników, takie działanie nie ma wpływu na poruszanie się. Zakres możliwych do wykonania ruchów przez robota prezentuje Ilustracja 6.4.

### 6.3 Miara skuteczności inteligentnego agenta

Do zbadania wydajności działania agenta konieczne jest zmierzenie pomiaru jego skuteczności. Określają ją kryteria, którym podlega jego zachowanie do oceny jego funkcjonowania. Kryterium takim jest zmieniający się stan środowiska, pod wpływem którego generowane są różne sekwencje zachowań agenta. W tym celu przeprowadzony został szereg symulacji, w których to zmierzony został poziom skuteczności działań agenta w oparciu o parametry symulowanego środowiska.

Parametry symulacji:

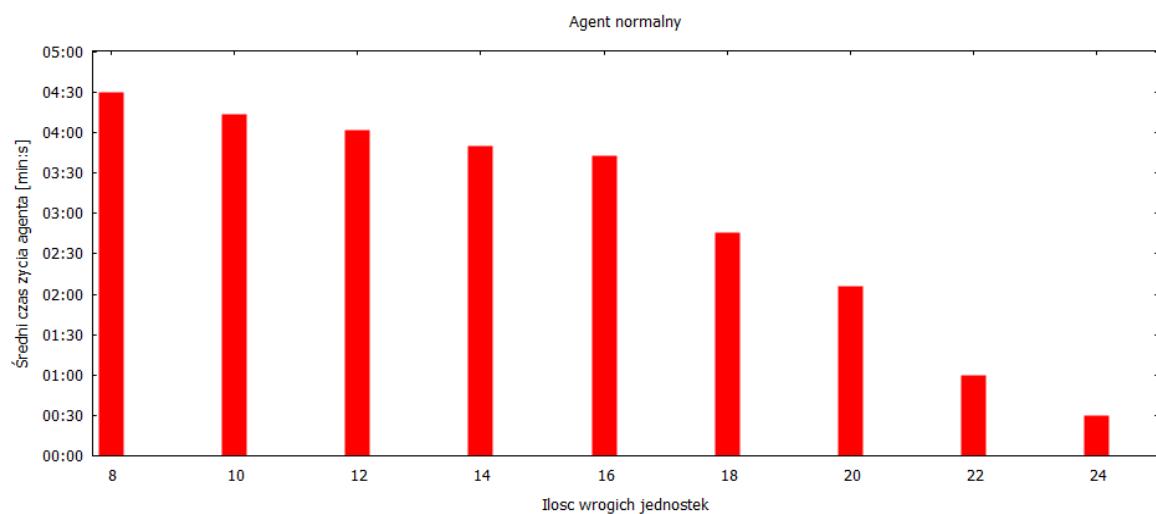
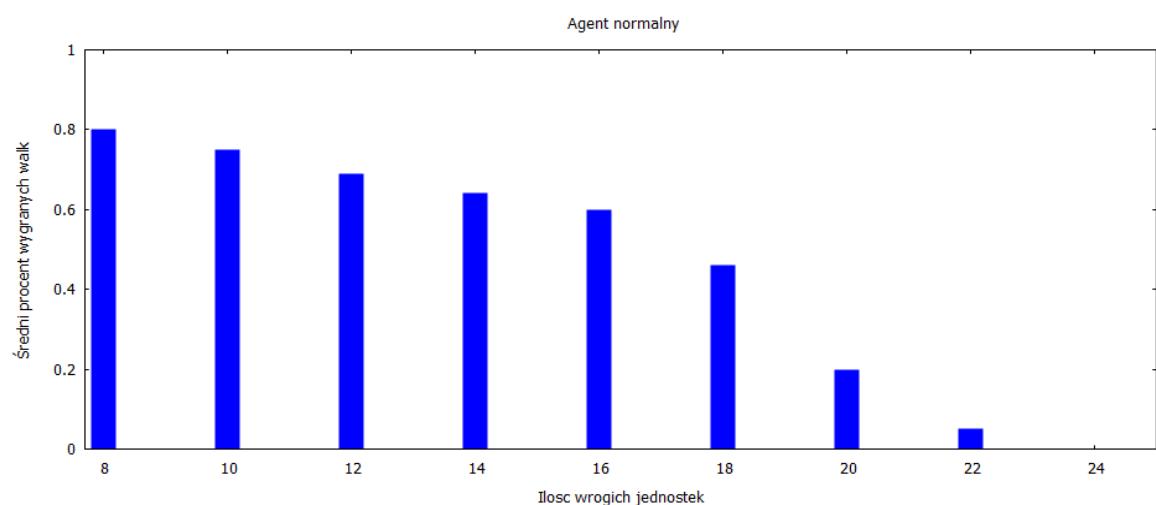
- Czas trwania symulacji jest równy 5 minut;
- Zadaniem agenta jest obrona statuy oraz pozostałe przy życiu;
- Agent wygrywa, jeśli wyeliminuje wszystkie wrogie jednostki, lub utrzyma statuę przy życiu przez czas trwania symulacji;
- Agent przegrywa, jeśli statua lub on zostanie zniszczony;
- Mierzony jest czas życia agenta w każdej symulacji, ilość wygranych, oraz poziom życia statuy;
- Liczba wrogich jednostek w kolejnych symulacjach jest zmienna.

Zmienny stan środowiska określa liczba wrogich czołgów znajdujących się na mapie. Przeprowadzone zostało po kilkanaście symulacji dla każdej z ilości przeciwników od 8 do 24. Dodatkowo do gry wprowadziliśmy czynnik mający wpływ na osobowość agenta. Stany, w podziale maszyny stanów określającej cel robota, wyliczają poziom aktywacji w sposób zaprezentowany w poprzednim podrozdziale. Tak uzyskane poziomy aktywacji są przemnażane dla danego stanu przez wartość odpowiedniej wagi mieszczącej się w zakresie (0,1). W ten sposób można faworyzować wybrane stany. Na tej podstawie stworzone zostały cztery różne osobowości agenta, które prezentuje Tabela 8.

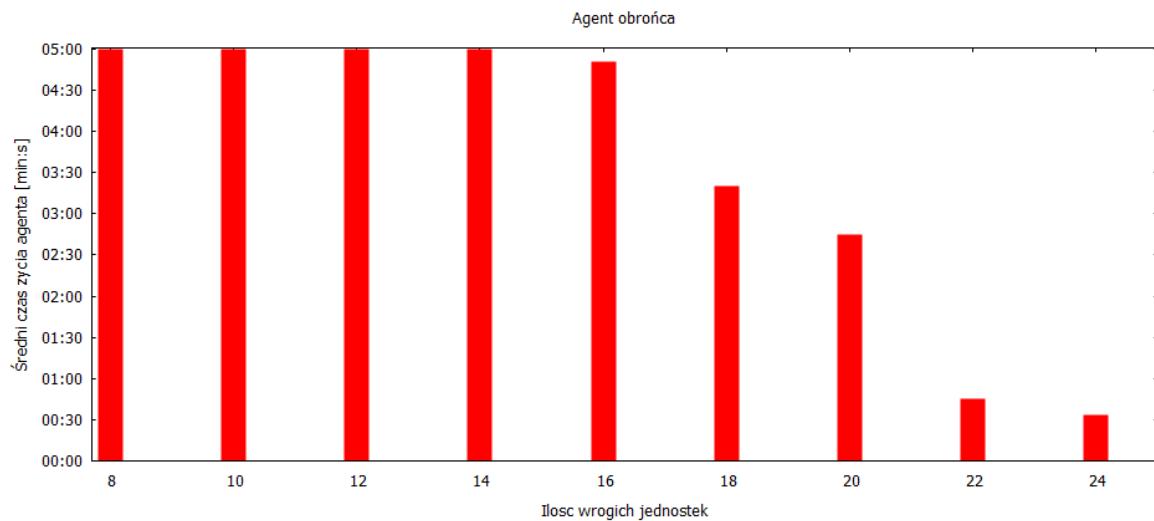
Tabela 8 Zestawienie różnych osobowości intelligentnego agenta.<sup>159</sup>

Agent	Normalny	Obrońca	Agresywny	Tchórz
Zabranie Apteczki	50%	20%	30%	100%
Obrona Statuy	50%	100%	30%	10%
Atak na Przeciwnika	50%	50%	100%	10%

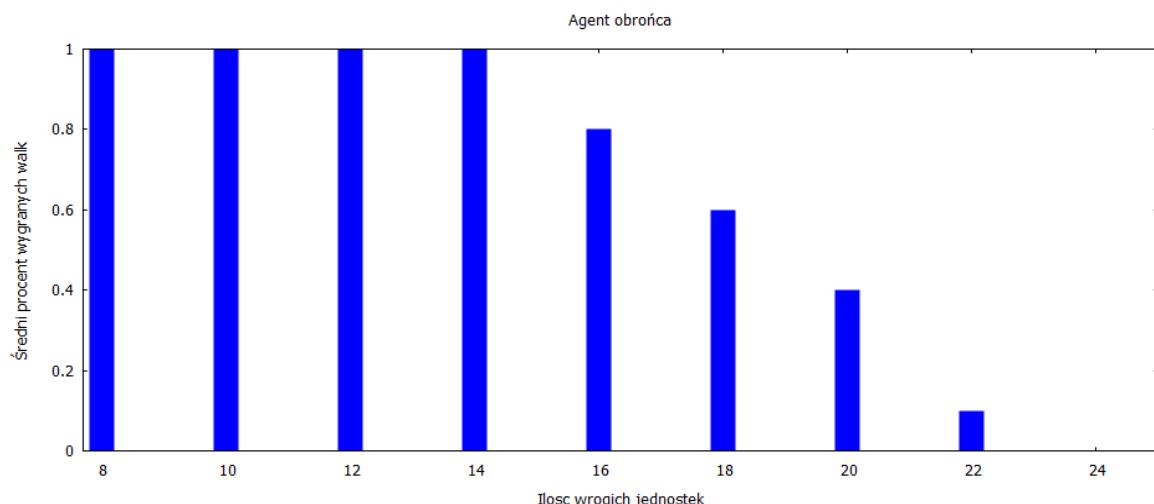
"Agent normalny", dla którego odpowiednie wagi są równe, czyli na wynikowe działanie ma wpływ wyłącznie rezultat rozmytego wnioskowania. "Agent obrońca" faworyzuje obronę statuy, kosztem własnego bezpieczeństwa i poziomu życia. Celem "agenta agresywnego" jest zlikwidowanie jak największej ilości przeciwników, a "agenta tchórz", pozostanie z wysokim poziomem własnego życia, bez względu na sytuację. Ilustracja 6.28 oraz Ilustracja 6.29 przedstawiają wyniki kilkudziesięciu symulacji dla agenta normalnego, jego średni czas życia oraz procent wygranych walk.

Ilustracja 6.28. Średni czas życia agenta normalnego w zależności od liczby wrogów.<sup>158</sup>Ilustracja 6.29. Średni procent wygranych walk agenta normalnego w zależności od ilości wrogich jednostek.<sup>158</sup><sup>159</sup> Źródło własne.

Wyniki dla agenta normalnego są dość satysfakcjonujące dla małej ilości przeciwników. Wraz z ich wzrostem średni czas życia agenta, jak i procent wygranych walk zaczynają stopniowo spadać. Powyżej 18 przeciwników agent nie radzi sobie z utrzymaniem siebie jak i statuy przy życiu. Ilustracja 6.30 oraz Ilustracja 6.31 przedstawiają wyniki symulacji dla agenta obrońcy:



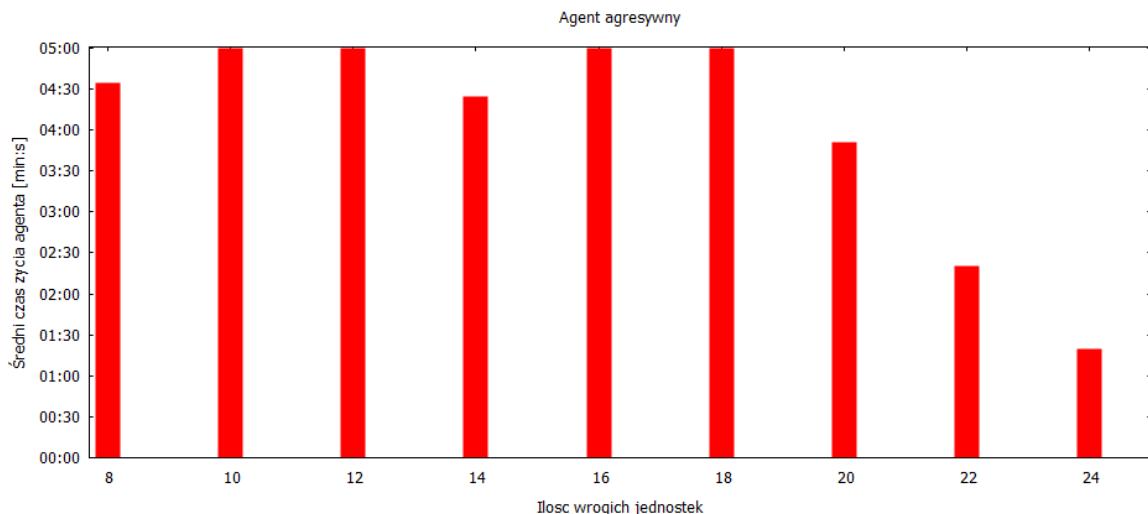
Ilustracja 6.30. Średni czas życia agenta obrońcy w zależności od liczby wrogów.<sup>160</sup>



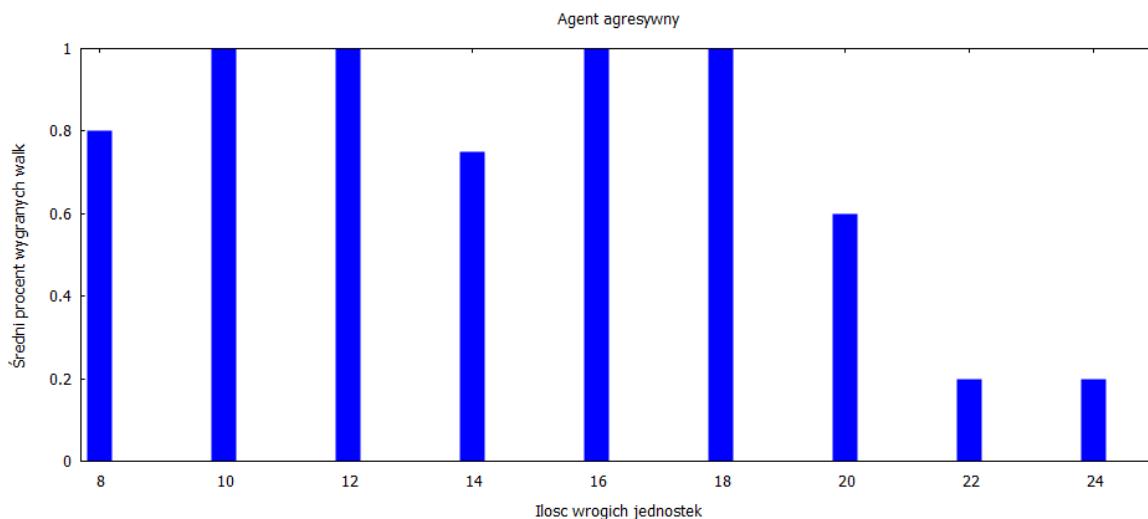
Ilustracja 6.31. Średni procent wygranych walk agenta obrońcy w zależności od ilości wrogich jednostek.<sup>159</sup>

Agent obrońcy, kosztem własnego bezpieczeństwa, poświęca więcej uwagi obronie statuy. Ta osobowość zapewnia mu 100% skuteczność dla symulacji do 14 wrogich czołgów. W pozostałych symulacjach wciąż radzi sobie lepiej niż agent normalny. Mimo niskiego stopnia faworyzacji stanu „Zbierania Apteczki”, agent jest w stanie dłużej utrzymać się przy życiu. Wciąż jednak dla współczynnika siły: on sam, przeciwko 24 wrogom, ani razu nie jest w stanie wygrać. Ilustracja 6.32 oraz Ilustracja 6.33 przedstawiają wyniki symulacji dla agenta agresywnego.

<sup>160</sup> Źródło własne.



Ilustracja 6.32. Średni czas życia agenta agresywnego w zależności od liczby wrogów.<sup>161</sup>

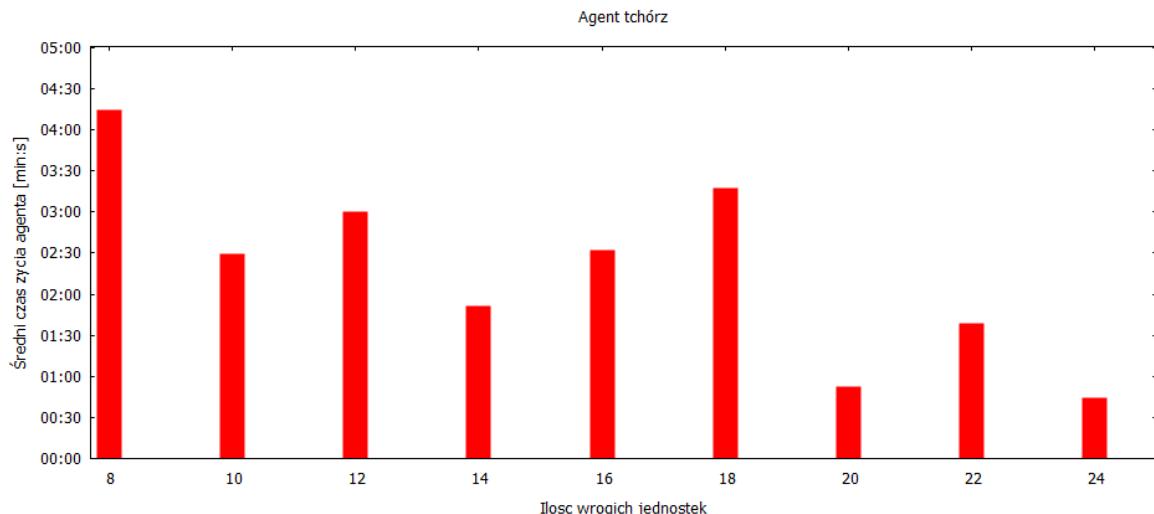


Ilustracja 6.33. Średni procent wygranych walk agenta agresywnego w zależności od ilości wrogich jednostek.<sup>160</sup>

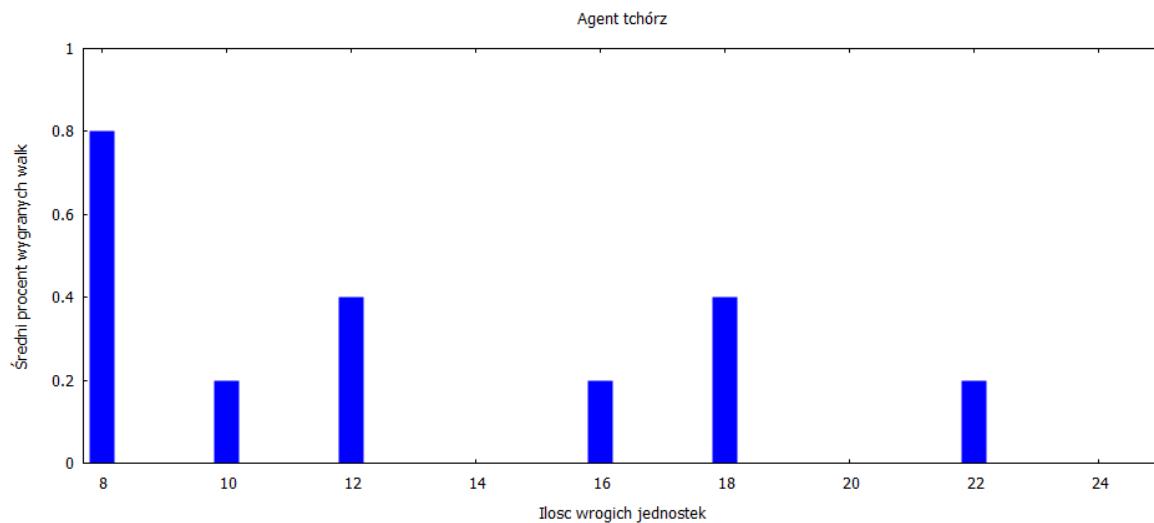
Wyniki agenta agresywnego charakteryzuje pewna fluktuacja. Jego faworyzowanym celem jest zniszczenie jak największej liczby przeciwników, co może skutkować długim pościgiem za wrogą jednostką, podczas gdy pozostałe niszczą statuetę. Takie działanie agenta jest ryzykowne. Dlatego też dla tylko 8, czy 14 wrogów, agent przegrywa. Odbija się to na sytuacji, gdy na terenie gry znajduje się gęsta ilość wrogich czołgów. Agent agresywny ryzykując, jest w stanie zniszczyć większość ilość czołgów i obronić statuetę. Dlatego też w 20% przeprowadzonych symulacji dla 22 i 24 wrogich jednostek, był w stanie wygrać. Ilustracja 6.34 oraz Ilustracja 6.35 przedstawiają wyniki symulacji dla agenta tchórza. Wynikająca z jego osobowości chęć przeżycia za wszelką cenę, a w szczególności za cenę życia statuy oraz walki z przeciwnikami, odbija się na niesatysfakcjonujących wynikach symulacji. Agent chcąc utrzymać się przy życiu, zostawał bezbronną statuetę bez obrony. Jedyną pozytywną stroną takiego podejścia jest

<sup>161</sup> Źródło własne.

fakt, że agent utrzymujący wysoki poziom swojego życia, jest w stanie przetrwać przy znacznie przeważającej liczbie wrogich jednostek. W przeprowadzonych symulacjach dla



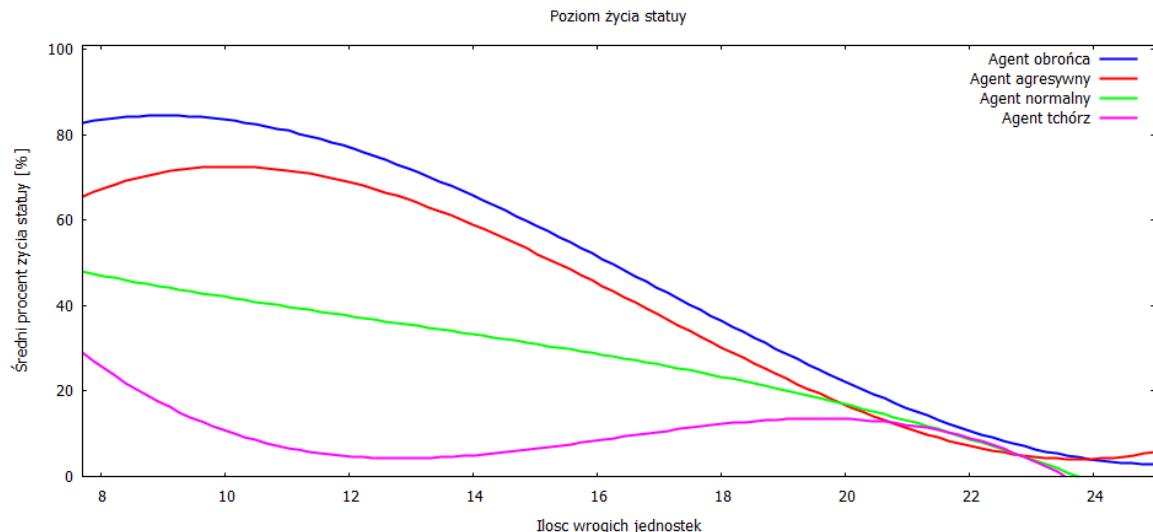
Ilustracja 6.34. Średni czas życia agenta tchorza w zależności od liczby wrogów.<sup>162</sup>



Ilustracja 6.35. Średni procent wygranych walk agenta tchorza w zależności od ilości wrogich jednostek.<sup>161</sup>

22 przeciwników, był w stanie w 20% przypadków wygrać bitwy. Ilustracja 6.36 przedstawia zestawienie wyników działania wszystkich agentów. Zobrazowane są wykresy średniego poziomu życia statuy, w zależności od ilości wrogich czołgów, dla wszystkich osobowości agenta. Najlepszą skuteczność, w utrzymywaniu statuy przy życiu wykazuje agent obrońca, drugie po nim, najlepsze osiągi ma agent agresywny. Skuteczność agenta normalnego, w utrzymywaniu statuy przy życiu, ma charakter liniowy, a agenta tchorza – opadającej sinusoidy. Zestawie to, odwzorowuje skuteczność działania poszczególnych agentów w realizacji ich głównego celu w danych warunkach środowiska.

<sup>162</sup> Źródło własne.



Ilustracja 6.36. Zestawienie średniego procentu poziomu życia statuy, w zależności od liczby wrogów, dla wszystkich agentów.<sup>163</sup>

Porównanie i analiza poszczególnych osobowości inteligentnego agenta, pokazuje jak duży wpływ osobowości agenta jak i zmienny stan otaczającego go środowiska, ma wpływ na skuteczność w jego działaniu. Dla tak zdefiniowanego środowiska i jego kryterium zmienności stanu – zmiennej ilości wrogich jednostek, bez problemu można ustalić listę najbardziej skutecznych agentów, jak i dostrzec wpływy wprowadzenia osobowości dla agentów. Dają one możliwość sterowania cechami agenta oraz tworzenie własnych profili jego działania, poprzez zewnętrzną ingerencję, bez modyfikacji bazy reguł rozmytych jak i systemu rozmytego wnioskowania. Idąc krok dalej, możemy zostawić funkcję modyfikacji osobowości samemu agentowi. Agent posiadający swój system percepcji, rozumowania, swoje cele, funkcje użyteczności, podejmujący decyzje, mógłby mieć wpływ na zmianę swojego działania poprzez uczenie się, dostosowując swoją osobowość do aktualnego stanu środowiska, w którym się znajduje.

<sup>163</sup> Źródło własne.

## 7. Podsumowanie i wnioski

Celem pracy było zaprojektowanie i implementacja Sztucznej Inteligencji robota w grze komputerowej z wykorzystaniem nowoczesnej grafiki komputerowej. Zasadniczą częścią pracy było stworzenie graficznej reprezentacji zaprojektowanej gry, oraz projekt i wdrożenie modelu Sztucznej Inteligencji.

Przed realizacją tematu pracy, istotnym elementem było zapoznanie się z aspektami tworzenia nowoczesnej grafiki komputerowej czasu rzeczywistego oraz teorią Sztucznej Inteligencji i logiki rozmytej. Podstawowym źródłem informacji, była wiedza przekazana nam przez profesorów Uniwersytetu Technicznego w Lizbonie. Główną rolę odegrała znajomość książki S. Russell, P. Norvig. „*Artificial Intelligence: A Modern Approach*”, uznawanej za jedną z podstawowych i najpopularniejszych w dziedzinie Sztucznej Inteligencji. Teorię jak i praktyczne zastosowanie logiki rozmytej przedłożył nam profesor J. M. C. Sousa, na podstawie jego książki „*Fuzzy Decision Making in Modeling and Control*”. Przygotowany przez nas opis teoretyczny, wraz z zawartą wiedzą, umożliwia dalszy rozwój zastosowania logiki rozmytej, jako głównego systemu decyzyjnego intelligentnego agenta.

Wybór środowiska programistycznego XNA Game Studio umożliwił nam stworzenie i implementacje realistycznej, trójwymiarowej gry czasu rzeczywistego. Duże wsparcie i dostępność wiedzy ze strony twórców oprogramowania, zapewniło nam szybkie zrozumienie jego architektury jak i szerokiego spektrum zastosowania. Wykorzystany przez nas silnik fizyki BEPU, kompatybilny z oprogramowanie XNA, posiadający obszerną bibliotekę algebraiczną i fizyczną, pozwolił nam stworzyć system oddziaływań fizycznych dla obiektów w grze. Do tworzenia prostych modeli na potrzeby naszej gry, wykorzystane zostało oprogramowanie SoftImage XSI. Umożliwia ono łatwy eksport obiektów geometrycznych do środowiska gry.

Rezultat naszej pracy oceniamy jako bardzo dobry, spełnione zostały wszystkie założenia projektowe gry. Osiągnięty efekt wizualny jest satysfakcjonujący według postawionych sobie przez nas kryteriów. Odbiega jakością od poziomu komercyjnych gier, jednak nie to było głównym celem naszej pracy. Zaproponowany przez nas model Sztucznej Inteligencji z wykorzystaniem rozmytego wnioskowania jako modułu decyzyjnego intelligentnego agenta działa poprawnie, spełniając swoje zadanie. Agent zachowuje się racjonalnie, dając do osiągnięcia swoich celów. Zastosowana przez nas logika rozmyta charakteryzująca się prostotą oraz łatwość w użyciu, jest bardzo dobrym narzędziem do modelowania niepewności, co jest również przydatne w grach komputerowych, gdzie agent powinien mieć zdolność posiadania wyłącznie takiej

wiedzy, jaką posiada gracz. Przy odpowiednim zaprojektowaniu systemu, kilka prostych reguł może dać bardzo duży wachlarz możliwości. Wadą logiki rozmytej jest jej złożoność, która znacznie wzrasta wraz ze wzrostem zmiennych, które powodują znaczny wzrost możliwych reguł. Dlatego logika rozmyta w grach jest wykorzystywana głównie w prostych odseparowanych systemach, by można było ją pozostawić prostą i łatwą do modyfikacji.

Istnieje wiele kierunków rozwoju naszej pracy. Inteligentny agent mógłby zostać udoskonalany na wiele sposobów, zaczynając od wprowadzenia bardziej skomplikowanych reguł i większej liczby zmiennych, jednak tego typu systemy wymagają wiele pracy i testów. Innym kierunkiem jest rozbudowa struktury inteligentnego agenta o możliwość uczenia się, np. tak jak zasugerowaliśmy pod koniec rozdziału 6.3, wykorzystując zastosowany przez nas system osobowości, do dostosowania się przez agenta do aktualnego stanu środowiska i optymalizacji swojego działania. Jeszcze inną drogą rozwoju jest wprowadzenie systemu wieloagentowego, z komunikacją i współpracą agentów, realizujących wspólne cele. W tego typu projektach możliwości jest wiele, jednak zaprezentowana metoda wnioskowania na podstawie logiki rozmytej w obecnej formie spełniła swoje zadanie.

## Załączniki

- M. Grygiel, R. Jabłonowski „Agent based on Fuzzy Decision Making”, 29.01.2011

Praca wykonana przez nas pod opieką prof. João Miguel da Costa Sousa jako projekt końcowy z przedmiotu „Intelligent Decision and Control” podczas naszych półrocznych studiów na Uniwersytecie technicznym w Lizbonie.

Praca opiera się na prototype napisanej przez nas gry „Tank Wars”. Jej celem było zaprojektowanie i implementacja inteligentnego agenta, za którego decyzje odpowiada logika rozmyta. Stworzony przez nas podstawowy modelu Sztucznej Inteligencji ukierunkował nasze dalsze prace nad niniejszą pracą magisterską. Artykuł napisany jest w języku angielskim. Dziękujemy prof. J. M. C. Sousa za konsultacje, cenne wskazówki i rady, oraz za miłą atmosferę podczas zajęć i pisania projektu.

# Bibliografia

1. **A. Turing**, *Computing Machinery and Intelligence*, Mind LIX (236): 433–460, 1950.
2. **K. Adamska**, <http://lacina.info.pl>, *Lingua Latina Omnibus*, Słownik łaciński [07.05.2011].
3. **S. Russell, P. Norvig**. *Artificial Intelligence: A Modern Approach*, Prentice Hall, wydanie 2, 2003.
4. **E. Moore**, *Cramming more components onto integrated circuit*, Electronics Magazine 38, 1965.
5. **C. Crawford**, *The Art of Computer Game Design*, Columbus: McGraw-Hill, 1984.
6. **J. M. C. Sousa, U. Kaymak**, *Fuzzy Decision Making in Modeling and Control*, World Scientific, 2002.
7. **A. J. Champandard**, *Teaming up with Halo's AI: 42 Tricks To Assist Your Game*, <http://aigamedev.com/open/reviews/halo-ai/> [19.09.2011].
8. **StrategyWiki**, <http://strategywiki.org>, *Battle City*, [http://strategywiki.org/wiki/Battle\\_City](http://strategywiki.org/wiki/Battle_City) [07.05.2011].
9. **MSDN**, <http://msdn.microsoft.com/en-us/>, oficjalna dokumentacja Microsoft XNA Game Studio, <http://msdn.microsoft.com/en-us/library/bb200104.aspx> [08.05.2011].
10. **J. McCarthy**, *Dartmouth Proposal*, Dartmouth Conferences, 1955.
11. **F. Dul**, *Wprowadzenie do Sztucznej Inteligencji*, Politechnika Warszawska, 2007
12. **J.M.C.Sousa**, *Fuzzy sets*, Intelligent Control and Decision Making, Technical University of Lisbon, 2010
13. **L. Rutkowski**, *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, 2005
14. **B. Schwab**, *AI Game Engine Programming*, Course Technology, wydanie 2, 2009.
15. [http://en.wikipedia.org/wiki/Jean\\_Piaget](http://en.wikipedia.org/wiki/Jean_Piaget) [19.09.2011].
16. **M. Buckland**, *Programming Game AI by Example*, Wordware Publishing, 2005.
17. <http://pl.wikiquote.org/wiki/Cyceron> [19.09.2011].
18. **J. Wexler**, *Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future*, <http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf> [07.09.2011].

19. **M. Belis**, *Computer and Video Game History*,  
[http://inventors.about.com/library/inventors/blcomputer\\_videogames.htm](http://inventors.about.com/library/inventors/blcomputer_videogames.htm) [19.09.2011].
20. **I. Millington, J. Funge**, *Artificial Intelligence For Games*, Morgan Kaufmann, wydanie 2, 2009.
21. **B. Harrison**, *Competitive Game Players*, <http://isites.harvard.edu/fs/docs/icb.topic622960.files/lecture-07.pdf> Harvard, 30.09.2009.
22. **A. J. Champandard**, *Top 10 Most Influential AI Games*,  
<http://aigamedev.com/open/highlights/top-ai-games/> [12.09.2011].
23. **A. J. Champandard**, *Flanking Total Wars AI: 11 Tricks to Conquer for Your Game*,  
<http://aigamedev.com/open/reviews/total-war-ai/> [22.09.2011].
24. **Y. Li, P. Musilek, L. Wyard-Scott**, *Fuzzy Logic in Agent-Based Game Design*, University of Alberta, Edmonton Canada, 2004.
25. **S. Harnad**, *What's Wrong and Right About Searle's Chinese Room Argument?*, Oxford University Press, 2001.
26. **Y. Li, P. Musilek, L. Kurgan**, *Battlecity Revived: Game Design with BDI.net*, University of Alberta, Edmonton Canada, 2004.
27. **M. E. Bratman**, *Intentions, Plans and Practical Reasoning*, Londyn :Harvard University Press, 1987.
28. **T. Leonard**, *GDC 2003: Building an AI Sensory System: Examining The Design of Thief: The Dark Project*, [http://www.gamasutra.com/gdc2003/features/20030307/leonard\\_01.htm](http://www.gamasutra.com/gdc2003/features/20030307/leonard_01.htm) [07.09.2011].
29. **A. J. Champandard**, *Sneaking Behind Thief's AI: 14 Tricks To Steal For Your Game*,  
<http://aigamedev.com/reviews/thief-ai/> [17.09.2011].
30. **K. D. Forbus**, *Simulation and Modeling: Under the hood of The Sims*,  
[http://www.cs.northwestern.edu/~forbus/c95-gd/lectures/The\\_Sims\\_Under\\_the\\_Hood\\_files/v3\\_document.htm](http://www.cs.northwestern.edu/~forbus/c95-gd/lectures/The_Sims_Under_the_Hood_files/v3_document.htm) [01.09.2011].
31. **A. J. Champandard**, *Living with The Sims AI: 21 Tricks To Adopt for Your Game*,  
<http://aigamedev.com/reviews/the-sims-ai/> [15.09.2011].
32. **A. J. Champandard**, *Teaming up with Halo's AI: 42 Tricks To Assist Your Game*,  
<http://aigamedev.com/open/reviews/halo-ai/> [19.09.2011].
33. **D. Isla**, *Handling Complexity in the Halo 2 AI*,  
[http://www.gamasutra.com/gdc2005/features/20050311/isla\\_01.shtml](http://www.gamasutra.com/gdc2005/features/20050311/isla_01.shtml) [11.09.2011].
34. **A. J. Champandard**, *Assaulting F.E.A.R.'s AI : 29 Tricks to Arm Your Game*,  
<http://aigamedev.com/reviews/fear-ai/> [22.09.2011].

35. **J. Wexler**, *Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future*,  
<http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf> [07.09.2011].
36. **M. Dramiński**, *Algorytm indukcji reguł decyzyjnych w problemach klasyfikacji i wyboru cech w zadaniach wysokowymiarowych*,  
[http://www.ipipan.eu/staff/m.draminski/files/mdr\\_phd.v2.pdf](http://www.ipipan.eu/staff/m.draminski/files/mdr_phd.v2.pdf) [20.09.2011].
37. **C. WOnge**, *Interactive Tours*, <http://www.worldwidetelescope.org> [01.07.2011].
38. **Korporacja NVIDIA**, *Demystifying 3D Graphics*, <http://developer.nvidia.com> [19.07.2011].
39. **J. Madeiras Pereira**, *The 3D Graphics Rendering Pipeline*, Animação e Visualização Tridimensional, Lisboa, Portugal 2009/2010.
40. **A. Oneal**, *Silverlight 3D Graphics*, <http://aarononeal.info/> [20.07.2011].
41. **K. Myzia**, *GF8800GTX, 8800GTS - rewolucja czy postęp?*, <http://pcarena.pl> [24.07.2011].
42. **R. Nordby**, *Introduction to BEPU physics and physics simulation*, Bepu Entertainment, LLC [25.12.2009].
43. **Lawrence**, *XNA Collision Detection for 3D models*, Sharky's Blog, [09.01.2012].
44. [http://pl.wikiquote.org/wiki/Maria\\_Dąbrowska](http://pl.wikiquote.org/wiki/Maria_Dąbrowska) [19.09.2011].
45. **C.Vernon**, *Terrain Generation with a Heightmap*, Chad Vernon's Blog, [27.05.2011].
46. **MSDN**, <http://create.msdn.com>, *Collision with a Heightmap*, [http://create.msdn.com/en-US/education/catalog/sample/collision\\_3d\\_heightmap](http://create.msdn.com/en-US/education/catalog/sample/collision_3d_heightmap) [27.09.2010].
47. **Sidvind**, *Skybox tutorial*, [http://sidvind.com/wiki/Skybox\\_tutorial](http://sidvind.com/wiki/Skybox_tutorial) [27.07.2011].
48. **Artist-3D**, *Catholic cathedral*, [http://artist-3d.com/free\\_3d\\_models/dnm/model\\_disp.php?uid=830](http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=830) [19.06.2007].
49. **W. Łużny**, *Wstęp do nauki o polimerach*, Skrypty Uczelniane, AGH Kraków, 1999.
50. **D. Halliday, R. Resnick, J. Walker**, *Podstawy fizyki*, PWN, 2007.
51. **MSDN**, <http://create.msdn.com>, *Chase Camera*,  
<http://create.msdn.com/en-US/education/catalog/sample/chasecamera> [26.11.2010].
52. **Artist-3D**, *Eagle Statue*, [http://artist-3d.com/free\\_3d\\_models/dnm/model\\_disp.php?uid=1647](http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=1647) [08.01.2010]
53. <http://pl.wikiquote.org/wiki/Platon> [20.09.2011].
54. **A. J. Champandard**, *10 Reasons The Age of Finite State Machines is Over*,  
<http://aigamedev.com/open/articles/fsm-age-is-over/> [20.09.2011].

55. **A. J. Champandard**, *Common Ways To Implement Finite State Machines In Games*,  
<http://aigamedev.com/articles/fsm-implementation/> [16.09.2011].
56. **Asger Feldthaus**, *XNA Procedural LTrees*, CodePlex, <http://ltrees.codeplex.com/>  
[01.12.2010]
57. **MSDN**, <http://create.msdn.com>, *Particle3D*,  
[http://create.msdn.com/en-US/education/catalog/sample/particle\\_3d](http://create.msdn.com/en-US/education/catalog/sample/particle_3d) [26.11.2010].

# Spis ilustracji

Ilustracja 1.1. Inteligentny agent .....	11
Ilustracja 2.1. Interakcja agenta ze środowiskiem.....	18
Ilustracja 2.2. Przykład agenta odkurzacza.....	19
Ilustracja 2.3. Struktura agenta refleksyjnego.....	23
Ilustracja 2.4. Struktura agenta refleksyjnego z modelem na świat.....	24
Ilustracja 2.5. Struktura agenta celowego.....	24
Ilustracja 2.6. Struktura agenta z funkcją użyteczności.....	25
Ilustracja 2.7. Struktura agenta uczącego się.....	26
Ilustracja 2.8. Precyza przeciwko Trafność – znacznie i wymowa w logice rozmytej.....	27
Ilustracja 2.9. Gaussowska funkcja przynależności.....	29
Ilustracja 2.10. Funkcja przynależności typu dzwonowego.....	30
Ilustracja 2.11. Funkcja przynależności klasy s.....	30
Ilustracja 2.12. Funkcja przynależności klasy $\pi$ .....	31
Ilustracja 2.13. Funkcja przynależności klasy $\gamma$ .....	31
Ilustracja 2.14. Funkcja przynależności klasy t.....	31
Ilustracja 2.15. Funkcja przynależności klasy L.....	32
Ilustracja 2.16. Zbiór B zawiera zbiór A.....	33
Ilustracja 2.17. Przekroje $\alpha$ zbioru rozmytego A. ....	33
Ilustracja 2.18. Zbiór rozmyty wypukły.....	34
Ilustracja 2.19. Zbiór rozmyty wklesły. ....	34
Ilustracja 2.20. Graficzne przedstawienie działania operacji przecięcia zbiorów rozmytych. ....	35
Ilustracja 2.21. Graficzne przedstawienie działania operacji sumy zbiorów rozmytych. .	36
Ilustracja 2.22. Graficzne przedstawienie działania operacji iloczynu algebraicznego zbiorów rozmytych.....	36
Ilustracja 2.23. Graficzne przedstawienie działania operacji dopełnienia zbioru rozmytego. ....	37
Ilustracja 2.24. Przedstawienie operacji koncentracji oraz rozcieńczania zbioru rozmytego. ....	38
Ilustracja 3.1. Battle City .....	44
Ilustracja 3.2. Cykl wykonawczy BDI.net.....	45
Ilustracja 3.3. Wybrane komponenty projektu „BattleCity.net” .....	46
Ilustracja 3.4. Wartości zmiennych rozmytych do kontroli zasięgu czołgów .....	47
Ilustracja 3.5 Zrzut z ekranu z gry – „sytuacja awaryjna”.....	47
Ilustracja 3.6. Straż w pokoju w grze Thief. [22] .....	49

---

Ilustracja 3.7. Przykład działania zmysłów w grze Thief. ....	50
Ilustracja 3.8. Rodzina w domu w grze The Sims. [22].....	51
Ilustracja 3.9. Funkcje wag potrzeb agenta [30], wartości w zakresie (0,1).....	52
Ilustracja 3.10. Przykład skierowanego grafu acyklicznego zachowań.....	54
Ilustracja 3.11. Jedno ze stworzeń rzucające zakłecie w grze Black & White. [22] .....	56
Ilustracja 3.12. Drzewo decyzyjne powstałe na podstawie powyższej tabeli. ....	57
Ilustracja 4.1. Wizualizacja danych: nauka i astronomia .....	58
Ilustracja 4.2. Obiekty geometryczne w grafice trójwymiarowej .....	59
Ilustracja 4.3. Wierzchołki i ich podstawowe konstrukcje.....	59
Ilustracja 4.4. Lewo i prawo skrętny układ współrzędnych .....	59
Ilustracja 4.5. Architektura potoku .....	60
Ilustracja 4.6. Architektura potoku – Vertex Shader oraz Pixel Shader.....	60
Ilustracja 4.7. Architektura potoku – Podetapy Geometrii.....	61
Ilustracja 4.8. Przykład zastosowania geometrii chwilowej – skupisko żołnierzy .....	62
Ilustracja 4.9. Kamera w przestrzeni świata .....	62
Ilustracja 4.10. Rodzaje powierzchni – z prawej gładka, z lewej chropowata.....	64
Ilustracja 4.11. Model oświetlenia Phonga.....	64
Ilustracja 4.12. Rodzaje projekcji, z lewej równoległa, z prawej perspektywiczna .....	67
Ilustracja 4.13. Przestrzeń ortogonalna .....	67
Ilustracja 4.14. Ostrosłup widzenia w przestrzeni perspektywicznej.....	67
Ilustracja 4.15. Transformacja przycinania sceny .....	68
Ilustracja 4.16. Mapowanie ekranu .....	69
Ilustracja 4.17. Architektura potoku – rasteryzacja.....	69
Ilustracja 4.18.Z lewej pudełko brzegowe, z prawej kule brzegowe [43] .....	71
Ilustracja 4.19. Osiem tysięcy orbitujących sześcianów .....	72
Ilustracja 5.1. Szkielet siatki geometrycznej tworzącej górzysty teren. [45] .....	75
Ilustracja 5.2. Położenie czołgu – widok z góry.....	75
Ilustracja 5.3. Położenie czołgu – widok z boku.....	76
Ilustracja 5.4. Sześciyan reprezentujący tło.....	76
Ilustracja 5.5. Schemat hierarchii geometrii czołgu.....	77
Ilustracja 5.6. Schemat systemu cząsteczek .....	80
Ilustracja 5.7. Struktura fizyki w grze. ....	80
Ilustracja 5.8. Podstawowy schemat budowy typowego silnika AI .....	82
Ilustracja 5.9. Diagram prostego automatu skończonego czerwonego duszka z opisywanej wcześniej gry Pac-Man. ....	83
Ilustracja 5.10. Typowy hierarchiczny automat skończony [55].....	84
Ilustracja 5.11. Przykładowy automat skończony statku kosmicznego.....	85
Ilustracja 5.12. Przykładowy rozmyty automat skończony tego samego statku kosmicznego.....	85
Ilustracja 5.13. Obliczanie poszukiwanego wektora.....	86

Ilustracja 5.14. Porównanie trasy zachowań podążania oraz przybycia, gdzie celem jest obiekt nieruchomy.....	87
Ilustracja 5.15. Obliczanie poszukiwanego wektora nakierowującego agenta na przewidziany punkt docelowy. ....	87
Ilustracja 5.16. Obszar detekcji przecinający obiekt w kształcie koła.....	88
Ilustracja 5.17. Kroki A, B i C. ....	89
Ilustracja 5.18. Składowe poszukiwanego wektora siły sterującej.....	89
Ilustracja 5.19. Unikanie ściany.....	90
Ilustracja 5.20. Rozmyty system wnioskujący.....	91
Ilustracja 5.21. Rozmyte wnioskowanie- reguły Mamdaniego.....	95
Ilustracja 5.22. Rozmyte wnioskowanie – reguły Larsena.....	96
Ilustracja 5.23. Rozmyte wnioskowanie – część wspólna sygnału wejściowego.....	96
Ilustracja 5.24. Rozmyte wnioskowanie – t-norma i iloczny. ....	97
Ilustracja 5.25. Rozmyte wyostrzanie – centrum średniej defuzyfikacji. ....	98
Ilustracja 5.26. Rozmyte wyostrzanie – środek ciężkości. ....	98
Ilustracja 5.27. Rozmyte wyostrzanie – maksimum funkcji przynależności. ....	99
Ilustracja 6.1. Bitmapa i odpowiadający jej teren w grze „ <i>Tank Wars</i> ”. .....	101
Ilustracja 6.2. Czołg na wznieśieniu. ....	101
Ilustracja 6.3. Różne widoki z kamery pościgowej.....	102
Ilustracja 6.4. Model czołgu i jego animacja.....	103
Ilustracja 6.5. Dwa rodzaje tekstury czołgu. ....	103
Ilustracja 6.6. Model kościoła. [48].....	104
Ilustracja 6.7. Konstrukcja modelu ściany.....	104
Ilustracja 6.8. Automatycznie generowane różne modele drzew. [56].....	105
Ilustracja 6.9. Symulacja cząsteczek przy wystrzale i trafieniu.....	105
Ilustracja 6.10. Bitmapy cząsteczek dymu, wybuchu i ognia. ....	106
Ilustracja 6.11. Działanie emitera cząsteczek przy wystrzale pocisku. ....	106
Ilustracja 6.12. Symulacją cząsteczek kurzu.....	106
Ilustracja 6.13. Bryły brzegowe otaczające obiekty w grze. ....	107
Ilustracja 6.14. Tocząca się po terenie kulista apteczka. ....	107
Ilustracja 6.15. Kolizja czołgu ze ścianą.....	108
Ilustracja 6.16. Eksplozja pocisku przy zderzeniu ze ścianą.....	108
Ilustracja 6.17. Schemat Maszyny Stanów sterującej robotem.....	109
Ilustracja 6.18. Diagram UML Maszyny Stanów sterującej robotem. ....	110
Ilustracja 6.19. Schemat Maszyny Stanów sterującej przeciwnikami robota.....	110
Ilustracja 6.20. Graficzne przedstawienie funkcji reprezentujących zdrowie robota. ....	111
Ilustracja 6.21. Graficzne przedstawienie funkcji reprezentujących poziom agresji.....	112
Ilustracja 6.22. Odwzorowanie rozmytego sygnału wejściowego na funkcje zdrowia A) Robota i B) jego przeciwnika.....	112
Ilustracja 6.23. Graficzna prezentacja przeprowadzonego wnioskowania przykładu 1..	115

---

Ilustracja 6.24. Odwzorowanie rozmytego sygnału wejściowego na funkcje zdrowia A) Robota i B) jego przeciwnika.....	116
Ilustracja 6.25. Graficzna prezentacja przeprowadzonego wnioskowania przykładowu 2..	118
Ilustracja 6.26. Funkcje mapujące odległość apteczki oraz statuy od robota. ....	119
Ilustracja 6.27. Funkcja $x2$ reprezentująca zniszczenia robota oraz statuy. ....	119
Ilustracja 6.28. Średni czas życia agenta normalnego w zależności od liczby wrogów ...	122
Ilustracja 6.29. Średni procent wygranych walk agenta normalnego w zależności od ilości wrogich jednostek.....	122
Ilustracja 6.30. Średni czas życia agenta obrońcy w zależności od liczby wrogów.....	123
Ilustracja 6.31. Średni procent wygranych walk agenta obrońcy w zależności od ilości wrogich jednostek.....	123
Ilustracja 6.32. Średni czas życia agenta agresywnego w zależności od liczby wrogów.	124
Ilustracja 6.33. Średni procent wygranych walk agenta agresywnego w zależności od ilości wrogich jednostek.....	124
Ilustracja 6.34. Średni czas życia agenta tchórza w zależności od liczby wrogów.....	125
Ilustracja 6.35. Średni procent wygranych walk agenta tchórza w zależności od ilości wrogich jednostek.....	125
Ilustracja 6.36. Zestawienie średniego procentu poziomu życia statuy, w zależności od liczby wrogów, dla wszystkich agentów. .....	126

# Podział pracy

1. Rozdział pierwszy:
  - Wprowadzenie, 1.2, 1.3 Rafał Jabłonowski,
  - 1.1 Michał Grygiel.
2. Sztuczna Inteligencja:
  - Michał Grygiel i Rafał Jabłonowski.
3. Sztuczna Inteligencja w grach:
  - 3.1, 3.3 Michał Grygiel,
  - 3.2 Rafał Jabłonowski.
4. Nowoczesna grafika komputerowa:
  - Rafał Jabłonowski.
5. Koncepcja inteligentnego i autonomicznego robota
  - 5.1, 5.2 Rafał Jabłonowski,
  - 5.3, 5.4 Michał Grygiel.
6. Realizacja
  - 6.1, 6.3 Rafał Jabłonowski,
  - 6.2 Michał Grygiel.
7. Podsumowanie i wnioski
  - Rafał Jabłonowski, Michał Grygiel.

## Zawartość DVD

Na płycie DVD zawarto materiały:

- Prezentacja ppt /Prezentacja,
- PDF z pracą magisterską /Praca magisterska,
- Pliki ilustracji /Praca magisterska – img,
- Pliki multimedialne /Praca magisterska – avi,
- Kod źródłowy gry „Tank Wars” /Gra
- Wersja instalacyjna gry „Tank Wars” /Gra – setup.
- Środowisko gry /Gra/Software
- Załączniki /Zalaczniki