Informatyka w Edukacji, XXII
UMK Toruń, 2026

BUDUJEMY MĄDREGO WROGA! PRAKTYCZNE WPROWADZENIE DO AI POPRZEZ PROGRAMOWANIE GIER W PYTHONIE

Damian Kurpiewski
Wydział Matematyki i Informatyki
Uniwersytetu Mikołaja Kopernika w Toruniu,
Instytut Podstaw Informatyki Polskiej Akademii Nauk
blackbat@mat.umk.pl; blackbat13.github.io

Abstract. *This article introduces artificial intelligence through a simple stealth game in Python and Pygame Zero. By designing an enemy that reacts to the player, changes states, and follows modifiable rules, students explore AI as decision making, perception, goal-oriented behavior, and agent-based systems. Experiments with patrol, pursuit, guarding, and prediction combine programming, game mechanics, geometry, and reflection on intelligent behavior in a classroom-friendly activity.*

1. Wstęp

Sztuczna inteligencja bywa dziś najczęściej kojarzona z dużymi modelami językowymi i czatbotami, ponieważ to właśnie one stały się najbardziej widocznym doświadczeniem AI w codziennym życiu uczniów. Takie skojarzenie jest jednak tylko fragmentem znacznie szerszego obrazu. Sztuczna inteligencja obejmuje również systemy podejmujące decyzje, algorytmy przeszukiwania, uczenie ze wzmocnieniem, agentów działających w środowisku niepełnej informacji oraz rozwiązania stosowane od lat w grach komputerowych [1]. W tym artykule proponuję praktyczną drogę do odkrycia tych mniej oczywistych oblicz sztucznej inteligencji: zbudowanie prostej gry w Pythonie, w której przeciwnik zachowuje się „inteligentnie", analizuje sytuację i wybiera działania zgodnie z przyjętą strategią. Taki projekt pozwala uczniom nie tylko rozwijać umiejętności programistyczne, lecz także zrozumieć, że AI to coś więcej niż ChatGPT. AI może oznaczać modelowanie zachowań, planowanie ruchów i reagowanie na zmieniający się stan świata. Dzięki takiemu podejściu programowanie staje się eksperymentem, a gra — bezpiecznym laboratorium, w którym można obserwować, jak proste reguły i algorytmy tworzą wrażenie rozumnego działania. Wpisuje się to również w zalecenia

dotyczące rozwijania kompetencji uczniów w zakresie rozumienia, stosowania i krytycznej oceny systemów AI [2].

2. Gra z inteligentnym strażnikiem

Projekt opisany w dalszej części artykułu polega na przygotowaniu prostej gry skradankowej w języku Python z wykorzystaniem biblioteki Pygame Zero. Na planszy znajdują się trzy główne obiekty: gracz, strażnik oraz skarb. Zadaniem gracza jest dotarcie do skarbu bez kontaktu ze strażnikiem. Pozornie jest to bardzo prosta mechanika, ale właśnie dzięki tej prostocie dobrze nadaje się do rozmowy o sztucznej inteligencji. Uczniowie mogą łatwo zauważyć, że „inteligencja" przeciwnika nie musi oznaczać rozmowy z człowiekiem ani generowania tekstu. Może polegać na obserwowaniu otoczenia, wykrywaniu gracza, przełączaniu się między stanami i wybieraniu działania odpowiedniego do sytuacji.

W omawianej grze strażnik zachowuje się inaczej w zależności od tego, czy widzi gracza. Gdy gracz znajduje się w zasięgu wzroku, przeciwnik zaczyna go ścigać. Gdy gracz jest poza zasięgiem, wykonywana jest osobna funkcja opisująca domyślne zachowanie strażnika. To miejsce staje się najważniejszym punktem eksperymentowania: można sprawić, aby strażnik stał w miejscu, patrolował wybrany obszar, podążał w stronę skarbu albo próbował blokować drogę graczowi. W ten sposób uczniowie otrzymują przestrzeń do własnych modyfikacji.

2.1. Przygotowanie środowiska

Do wykonania projektu potrzebny jest zainstalowany Python oraz biblioteka Pygame Zero. W najprostszym wariancie można ją doinstalować poleceniem *pip install pgzero*. Oprócz tego należy przygotować katalog images z grafikami wykorzystywanymi przez grę, na przykład: *player.png*, *guard1.png*, *guard2.png*, *guard3.png* oraz *treasure.png*. Nazwy plików są istotne, ponieważ Pygame Zero odwołuje się do obrazów przez ich nazwy.

Program można umieścić w pliku *main.py*. Na początku importujemy moduł *math*, potrzebny później do obliczania odległości i kierunku ruchu, oraz *pgzrun*, który pozwala uruchomić grę Pygame Zero jak zwykły skrypt Pythona. Następnie definiujemy rozmiar okna, tytuł gry oraz najważniejsze stałe, takie jak szybkość gracza, szybkość strażnika i promień widzenia przeciwnika.

2.2. Aktorzy gry i podstawowe reguły

W Pygame Zero obiekty widoczne na ekranie można reprezentować za pomocą klasy Actor. W naszym projekcie tworzymy trzy takie obiekty: gracza umieszczonego po lewej stronie ekranu, strażnika znajdującego się na środku oraz skarb po prawej stronie.

Taki układ od razu wprowadza napięcie: gracz chce przejść z jednej strony na drugą, a strażnik może mu w tym przeszkodzić.

Do obiektu gracza można dopisać dodatkową informację o stanie gry, na przykład z wartością *„playing”*, *„win”* albo *„lose”*. Dzięki temu program wie, czy gra wciąż trwa, czy gracz dotarł do skarbu, czy został złapany. Warunek wygranej jest prosty: gracz musi zetknąć się ze skarbem. Warunek przegranej również jest intuicyjny: jeśli gracz zetknie się ze strażnikiem, zostaje złapany. Ta prostota pomaga skupić uwagę uczniów na zachowaniu przeciwnika, czyli na najważniejszym elemencie projektu.

2.3. Sztuczna inteligencja przeciwnika

Najważniejszą częścią programu jest funkcja opisująca zachowanie strażnika wtedy, gdy nie widzi on gracza. W wersji podstawowej strażnik porusza się w stronę skarbu. Można to potraktować jako bardzo prostą strategię: przeciwnik nie wie, gdzie znajduje się gracz, więc pilnuje celu, do którego gracz prawdopodobnie będzie zmierzał. Z punktu widzenia ucznia jest to dobry przykład myślenia algorytmicznego: nie zapisujemy „prawdziwej inteligencji”, ale projektujemy regułę, która w określonych warunkach daje sensowne zachowanie.

Drugim ważnym elementem jest funkcja sprawdzająca, czy strażnik widzi gracza. Możemy w tym celu zastosować prostą regułę: jeśli odległość między strażnikiem a graczem jest mniejsza lub równa ustalonemu promieniowi widzenia, strażnik uznaje, że gracz został zauważony. Nie jest to realistyczne pole widzenia, ale świetnie pokazuje podstawową ideę percepcji agenta. Agent, czyli w tym przypadku strażnik, odbiera informację o świecie i na tej podstawie wybiera działanie.

Tabela 1. Stany strażnika i jego zachowanie

Stan strażnika	Warunek	Zachowanie
Alarm	Gracz znajduje się w promieniu widzenia	Strażnik porusza się w stronę gracza
Patrol lub spoczynek	Gracz jest poza promieniem widzenia	Strażnik wykonuje funkcję opisującą zachowanie domyślne

Logikę strażnika można opisać jako prostą maszynę stanów. W pierwszym stanie, nazwijmy go stanem alarmu, gracz znajduje się w zasięgu widzenia i strażnik podąża bezpośrednio w jego stronę. W drugim stanie, spoczynku lub patrolu, gracz nie jest widoczny i uruchamiana jest funkcja zachowania domyślnego. Uczniowie mogą dzięki

temu zobaczyć, że wiele systemów AI w grach nie opiera się na jednym „magicznym” algorytmie, lecz na przejrzystych regułach przełączania między zachowaniami.

2.4. Ruch, odległość i kierunek

Aby strażnik mógł poruszać się w stronę gracza albo skarbu, potrzebna jest funkcja przesuwająca wybranego aktora w kierunku punktu docelowego. Najpierw obliczamy różnicę między współrzędnymi celu i aktualną pozycją aktora. Następnie wyznaczamy odległość w linii prostej, korzystając z funkcji *math.hypot*. Po podzieleniu składowych ruchu przez odległość otrzymujemy wektor o długości jeden, który można pomnożyć przez prędkość strażnika. Dzięki temu przeciwnik porusza się płynnie i z jednakową szybkością niezależnie od tego, czy idzie poziomo, pionowo czy po przekątnej.

Ten fragment jest dobrą okazją do połączenia informatyki z matematyką. Uczniowie widzą, że pojęcia takie jak odległość, wektor kierunku czy normalizacja nie są abstrakcyjnymi zapisami z podręcznika, lecz praktycznymi narzędziami potrzebnymi do stworzenia ruchu w grze. Co ważne, nie trzeba od razu wprowadzać formalnego języka algebry liniowej. Wystarczy intuicja: sprawdzamy, gdzie jest cel, w którą stronę trzeba iść, a następnie wykonujemy mały krok w tym kierunku.

2.5. Pętla gry i reakcje na zdarzenia

Pygame Zero automatycznie wywołuje funkcję *update()* wiele razy na sekundę. To właśnie tam umieszczamy logikę gry: odczyt klawiatury, przesuwanie gracza, decyzję strażnika oraz sprawdzanie warunków wygranej i przegranej. Gracz może poruszać się za pomocą strzałek lub klawiszy WASD. Strażnik natomiast najpierw sprawdza czy widzi gracza. Jeśli tak, zaczyna pościg. Jeśli nie, wykonuje swoje domyślne zachowanie.

Za rysowanie odpowiada funkcja *draw()*. Wypełnia ona tło, wyświetla skarb, gracza i strażnika oraz pokazuje komunikaty o wygranej lub przegranej. Warto zwrócić uwagę na zmianę grafiki strażnika: inny obraz jest używany, gdy przeciwnik patroluje, inny, gdy zauważył gracza, a jeszcze inny po złapaniu gracza. Dzięki temu uczniowie widzą nie tylko kodową, lecz także wizualną reprezentację stanów AI.

Ostatnim elementem podstawowej wersji gry jest obsługa klawisza restartu. Po naciśnięciu **R** gracz, strażnik i skarb wracają na pozycje początkowe, a stan gry ponownie przyjmuje wartość *playing*. Taki mechanizm jest szczególnie przydatny podczas zajęć, ponieważ pozwala szybko testować kolejne pomysły bez ponownego uruchamiania programu.

2.6. Eksperymenty z AI

Największa wartość edukacyjna projektu ujawnia się wtedy, gdy uczniowie zaczynają zmieniać funkcję odpowiedzialną za zachowanie strażnika. W najprostszym

wariancie można wpisać instrukcję *pass*, aby strażnik nic nie robił, gdy nie widzi gracza. Następnie można porównać ten wariant z przeciwnikiem, który pilnuje skarbu albo stale idzie w stronę gracza. Każda taka zmiana wpływa na poziom trudności gry i pozwala rozmawiać o tym, czym jest strategia przeciwnika.

Tabela 2. Pomysły na zachowanie strażnika

Pomysł	Efekt w grze	Powiązanie z AI
Brak ruchu	Strażnik reaguje tylko po zauważeniu gracza	Prosta reakcja warunkowa
Ruch w stronę skarbu	Strażnik chroni cel gracza	Zachowanie ukierunkowane na cel
Ruch w stronę gracza	Gra staje się trudniejsza i bardziej agresywna	Pościg za celem
Blokowanie drogi	Strażnik próbuje znaleźć się między graczem a skarbem	Proste planowanie pozycji
Przewidywanie ruchu	Strażnik celuje w miejsce, do którego gracz prawdopodobnie zmierza	Predykcja na podstawie poprzedniego ruchu

Takie eksperymenty można prowadzić indywidualnie albo w grupach. Każdy zespół może zaproponować własną wersję strażnika, a następnie porównać, która strategia jest najtrudniejsza, najbardziej przewidywalna albo najbardziej „sprawiedliwa” dla gracza. W ten sposób zajęcia naturalnie prowadzą do rozmowy o projektowaniu systemów AI: o tym, że zachowanie inteligentne zależy od celu, informacji dostępnej dla agenta i reguł podejmowania decyzji.

3. Wnioski

Budowa prostej gry z przeciwnikiem sterowanym regułami jest dobrym sposobem na odczarowanie pojęcia sztucznej inteligencji. Uczniowie widzą, że AI nie musi być odległą technologią dostępną tylko w dużych systemach ani wyłącznie narzędziem do generowania tekstu. Może być zbiorem prostych, zrozumiałych decyzji zapisanych w kodzie, które razem tworzą zachowanie odbierane jako inteligentne.

Projekt ma również tę zaletę, że łączy kilka ważnych obszarów kształcenia informatycznego: programowanie w Pythonie, pracę z pętlą gry, geometrię ruchu, projektowanie reguł oraz testowanie hipotez. Zmiana jednej funkcji natychmiast prowadzi do

widocznej zmiany zachowania przeciwnika, co sprzyja uczeniu się przez eksperymentowanie. Dzięki temu uczniowie mogą nie tylko poznać podstawy programowania gier, lecz także zrozumieć, że sztuczna inteligencja ma wiele oblicz — od prostych automatów decyzyjnych po zaawansowane modele uczące się na danych.

Pełen kod gotowej gry wraz z dodatkowymi rozszerzeniami dostępny jest w serwisie GitHub [3].

Literatura

1. Russell S., Norvig P., *Artificial Intelligence: A Modern Approach*, 4th Global Edition, Pearson, 2021. Dostępny opis zakresu podręcznika: https://aima.eecs.berkeley.edu/~russell/aima/global-index.html.
2. Miao F., Shiohira K., Lao N., *AI competency framework for students*, UNESCO, 2024.
3. https://github.com/blackbat13/PatrolPyGameZero.