

Metoda sztucznego potencjału

Tomasz Fiechowski, Wiktor Kobiela

Wymagania

Program powinien symulować ruch wahadła w poprzek prostokątnego pomieszczenia oraz ruch robota, którego ścieżka planowana jest z wykorzystaniem metody sztucznego potencjału, którego zadaniem jest przejechać pomieszczenie wzdłuż.

- a. Szerokość wahadła musi być dwa razy większa niż szerokość robota.
- b. Robot powinien startować w losowym momencie czasu i omijać wahadło.

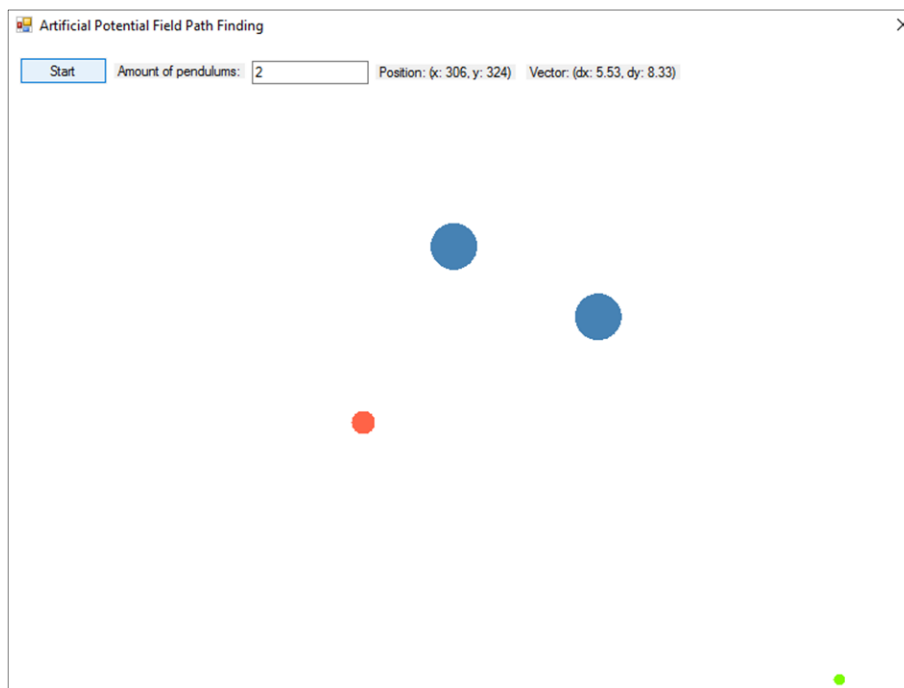
Ad a. W projekcie zaimplementowano możliwość ustawienia większej ilości wahadeł losowo rozmieszczonych w środkowym obszarze pomieszczenia. Dodatkowo, są one ustawiane pod losowym kątem oraz poruszają się z losową prędkością i mają losowe początkowe wychylenie.

Ad b. Robot zawsze startuje z tego samego miejsca, jednak ze względu na losowe rozmieszczenie wahadeł, możemy uznać, że startuje z różnych pozycji, spełniając wymóg b.

Użyte technologie

Projekt został zaimplementowany w języku C# z użyciem technologii WinForms. Technologia ta umożliwia bardzo prosty sposób rysowania kształtów 2D.

Interfejs użytkownika



Rys. 1. Interfejs użytkownika

Na rys. 1. Pokazany został interfejs użytkownika. W lewym górnym rogu widzimy przycisk „Start”, który służy do rozpoczęcia symulacji. Następnie dostępne jest pole tekstowe służące do wprowadzania ilości wahadła znajdujących się w pomieszczeniu. Dalej widać współrzędne oraz składowe wektora prędkości robota – są one aktualizowane na bieżąco podczas ruchu robota.




Symbol	Rola
	Robot
	Wahadło
	Punkt docelowy

Tabela 1. Wykaz symboli używanych podczas symulacji

Metoda sztucznego potencjału

Metoda sztucznego potencjału działa w następujący sposób: obiekty (przeszkody), które chcemy omijać, mają ujemny potencjał, przez co działają na robota siłą odpychającą. Z kolei punkt docelowy ma dodatni potencjał i działa „globalnie” na robota, przez co robot jest przyciągany do celu. W każdej chwili czasu podczas ruchu robota, liczony jest wypadkowy wektor sił działający na niego. W efekcie, po odpowiednim zaimplementowaniu takiego algorytmu, robot jest w stanie

omijać przeszkody i dotrzeć do celu nie znając ścieżki z góry, a jedynie reagując na chwilowe wartości wypadkowej potencjału, a co za tym idzie siły odpychającej albo przyciągającej.

Implementacja

W zaimplementowanym programie, robot w każdej chwili czasu liczy wypadkowy wektor przesunięcia na podstawie wielkości i odległości ładunków (potencjałów).

$$d_x = \frac{(o_x - r_x) \cdot o_c}{|v_{or}|^2}$$

Gdzie:

d_x - przesunięcie robota

o_x - współrzędna x położenia przeszkody (o – obstacle)

r_x - współrzędna x położenia robota (r – robot)

o_c - ładunek przeszkody

$|v_{or}|^2$ - kwadrat długości wektora pomiędzy robotem a przeszkodą

Warto zaznaczyć, że za wielkość ładunku robota przyjęto wartość +1.

Analogicznie postępujemy dla współrzędnych y otrzymując przesunięcie d_y .

$$d_y = \frac{(o_y - r_y) \cdot o_y}{|v_{or}|^2}$$

Takie przesunięcia liczymy dla każdej przeszkody i sumujemy ze sobą:

$$d_{xc} = \sum_{i=0}^N d_x(o_i), d_{yc} = \sum_{i=0}^N d_y(o_i)$$

Następnie tworzymy wektor jednostkowy na podstawie wartości d_{xc} oraz d_{yc} dzieląc je przez długość wektora skonstruowanego za ich pomocą:

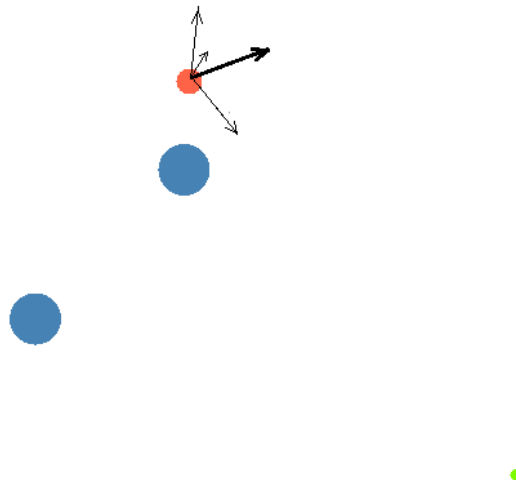
$$d_{xy} = \sqrt{d_{xc} * d_{xc} + d_{yc} * d_{yc}}$$

Dzielimy wartości d_{xc} oraz d_{yc} przez otrzymaną długość wektora oraz mnożymy je przez prędkość robota v_r .

$$D_x = \frac{d_{xc}}{d_{xy}} \cdot v_r, \quad D_y = \frac{d_{yc}}{d_{xy}} \cdot v_r$$

Otrzymane wartości D_x oraz D_y oznaczają przesunięcie robota do następnej pozycji.

Działanie programu



Rys. 2. Wektor wypadkowy sił działających na robota

Na rys. 2 widać sposób, w jaki obliczany jest wypadkowy wektor sił działających na robota. Długości wektorów w rzeczywistości są mniejsze, ale na potrzeby przykładu zostały odpowiednio wydłużone. Widzimy dwa wektory sił zwrócone ku górze, których źródłem są niebieskie wahadła. Wektor zwrócony do punktu docelowego przyciąga robota do celu. Wytluszczony, wypadkowy wektor określa następne położenie robota.

Po osiągnięciu przez robota punktu docelowego, wyświetlany jest komunikat o sukcesie:

