Inteligentna Śmieciarka

Projekt ze Sztucznej Inteligencji

1. TWORZENIE MAPY DYSKRETNEJ

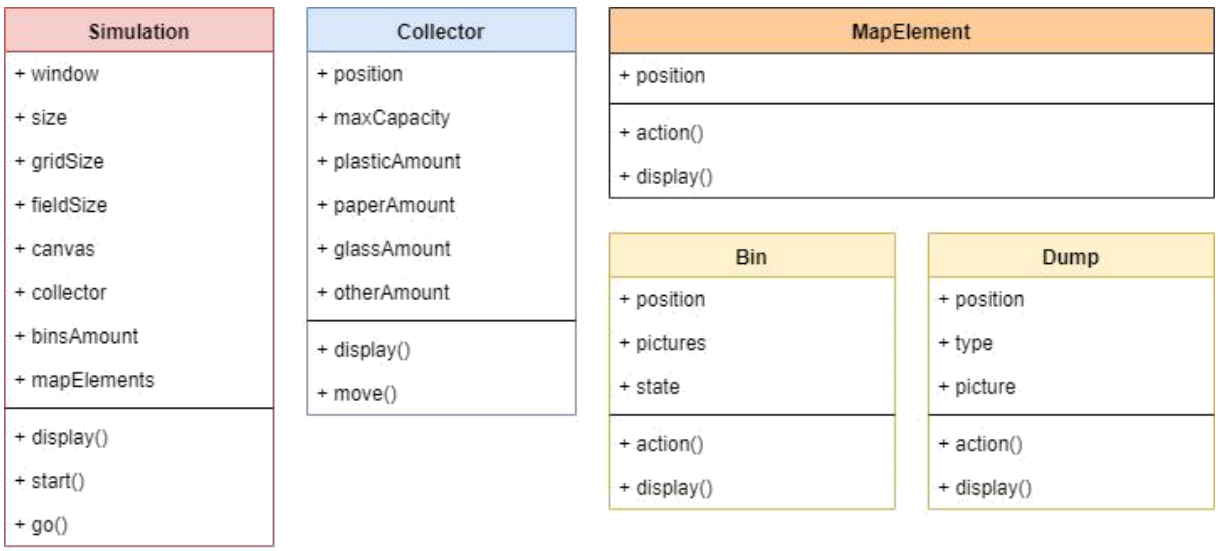
1.1 Informacje ogólne

Zgodnie z założeniami tematu projektu jakim jest stworzenie inteligentnej śmieciarki, nasza grupa obrała za pierwszy cel możliwość wygenerowania mapy dwuwymiarowej, na której znalazłyby się obiekty, takie jak:

- Drogi
- Śmietniki
- Śmieci

1.2 Podjęte działania i środowisko programistyczne

Do zrealizowania naszego planu posłużyliśmy się językiem programowania Python. Zaczęliśmy od stworzenia kilku klas, które odpowiadają obiektom na mapie, takie jak: śmieciarka, śmietniki, droga, itd. Następnie zainicjowaliśmy w konstruktorach tych obiektów ich pozycję oraz odpowiednie obrazki (została ona pobrana w całości z Google Grafika). Dodaliśmy również klasę symulacji, w której zawarliśmy obszar samej aplikacji okienkowej, a także całą logikę, która generuje nam poszczególne obiekty na mapie. Poza tym dodany został podstawowy ruch agenta – śmieciarki. Poniżej znajduje się grafika ilustrująca strukturę projektu:



2. ALGORYTM DFS (DEPTH-FIRST SEARCH)

2.1 Zdefiniowanie problemu

Celem zadania jest umożliwienie naszemu agentowi (śmieciarce) poruszania się, używając algorytmu przeszukiwania w głąb. Strategią tego algorytmu jest przechodzenie z wierzchołka początkowego, w naszym przypadku jest to pole w którym agent zaczyna swoją trasę, do pierwszych nieodwiedzonych pól, aż do momentu, kiedy nie ma możliwości dalszego poruszania się. Wtedy cofamy się do pola, z którego ostatnio przyszliśmy i przechodzimy do następnego nieodwiedzonego pola (o ile istnieje). Czynności te powtarzamy do momentu odwiedzenia wszystkich pól, co w naszym przypadku oznacza, że wyznaczyliśmy wszystkie drogi do danego punktu.

2.2 Rozwiązanie problemu

W celu uzyskania wyników, używamy tzw. *wrappera*, który uruchamia funkcję *dfs* odpowiedzialną za wyszukanie ścieżki. Funkcja *dfs* posiada w sobie stos, w którym zapisane ma współrzędne przebytych punktów. Na podstawie ostatniego elementu na stosie, sprawdzane są warunki możliwego przejścia do innych punktów. Jeżeli można iść dalej, to współrzędna zostaje wrzucona na stos. Cykl powtarza się. Za każdym razem gdy funkcja *dfs* zakończy swoje działanie, *wrapper* sprawdza czy warunki końcowe są spełnione. Jeżeli tak, to kończy swoje działanie.

3. ALGORYTM BFS (BREADTH-FIRST SEARCH)

2.1 Zdefiniowanie problemu

Celem zadania jest umożliwienie naszemu agentowi poruszania się, używając algorytmu przeszukiwania wszerz. Strategią tego algorytmu jest wyznaczenie wszystkich wierzchołków sąsiadujących z wierzchołkiem startowym, a następnie ich sprawdzanie. Cykl ten powtarzamy, aż do wyznaczenia wszystkich wierzchołków. Ważne jest, aby zapamiętać czy dany wierzchołek był już wcześniej odwiedzony, żeby uniknąć zapętlenia.

2.2

W celu uzyskania wyników, używamy *wrappera*, który uruchamia funkcję *bfs* odpowiedzialną za wyszukanie ścieżki. Funkcja *bfs* posiada w sobie kolejkę, w której zapisane ma współrzędne przebytych punktów. Na pierwszego elementu na liście, sprawdzane są warunki możliwego przejścia do innych punktów. Jeżeli można iść dalej, to współrzędna zostaje dołączona do kolejki. Cykl powtarza się do momentu sprawdzenia wszystkich możliwych punktów. Gdy funkcja *dfs* kończy swoje działanie, *wrapper* sprawdza czy warunki końcowe są spełnione. Jeśli tak jest, to kończy swoje działanie.