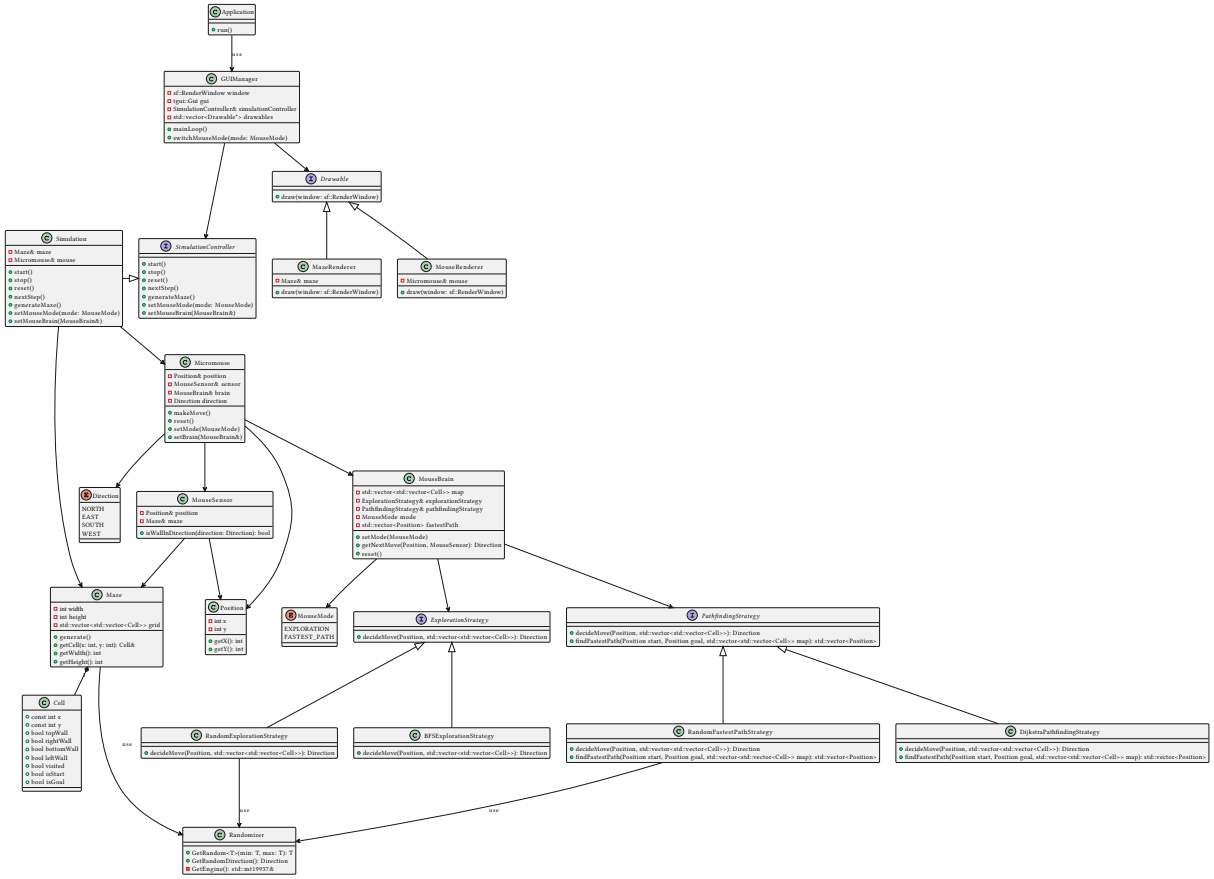
Zaawansowane C++

Diagram klas

Temat: **Micromouse**

Autor: **Jan Augustyn, 342201**

1. Diagram klas



2. Opis klas

2.1. Klasy podstawowe

Application

Główna klasa aplikacji. Tworzy symulację oraz uruchamia interfejs użytkownika.

SimulationController

Interfejs kontrolujący przebieg symulacji. Umożliwia uruchamianie, zatrzymywanie i resetowanie symulacji, a także zmianę trybu myszy oraz jej strategii.

Simulation

Reprezentuje symulację w programie. Przechowuje odniesienia do labiryntu i myszy, kontroluje ich działanie oraz umożliwia generacje labiryntu i zmianę strategii myszy.

2.2. Interfejs graficzny

GUIManager
Zarządza interfejsem użytkownika. Obsługuje okno aplikacji, elementy graficzne oraz interakcję użytkownika z symulacją.

Drawable
Interfejs dla obiektów rysowalnych. Każdy obiekt, który może być wyświetlany, implementuje metodę `draw()`.

MazeRenderer
Odpowiada za rysowanie labiryntu na ekranie.

MouseRenderer
Odpowiada za wizualizację myszy i jej ruchu w labiryncie.

2.3. Labirynt

Maze
Reprezentuje labirynt. Przechowuje jego rozmiar i układ ścian, umożliwia dostęp do poszczególnych komórek oraz losową generację labiryntu.

Cell
Pojedyncza komórka labiryntu. Zawiera między innymi informacje o swojej pozycji oraz obecności ścian w czterech kierunkach.

2.4. Mysz i algorytmy

Micromouse
Reprezentuje robota ‘mysz’. Odpowiada za ruch, przetwarzanie danych z sensorów oraz korzystanie z logiki decyzyjnej dostarczanej przez `MouseBrain`. Może resetować swoją pozycję oraz zmieniać strategię ruchu i trybu.

MouseSensor
Symuluje sensory myszy. Pozwala sprawdzić, czy w aktualnej pozycji myszy znajduje się ściana w danym kierunku.

Position
Prosta klasa reprezentująca współrzędne myszy w labiryncie.

Direction
Typ wyliczeniowy reprezentujący cztery możliwe kierunki ruchu: północ, południe, wschód i zachód.

MouseMode
Typ wyliczeniowy określający aktualny tryb działania myszy: eksploracja lub podążanie najkrótszą ścieżką.

MouseBrain
Przechowuje mapę labiryntu, zarządza strategiami eksploracji i znajdowania ścieżki oraz decyduje o kolejnych ruchach myszy.

2.5. Algorytmy eksploracji

ExplorationStrategy
Interfejs dla strategii eksploracji. Definiuje metodę `decideMove()`, która wybiera kolejny ruch myszy na podstawie mapy.

RandomExplorationStrategy
Strategia losowej eksploracji. Mysz porusza się w losowym, dostępnym kierunku.

BFSExplorationStrategy
Strategia eksploracji oparta na algorytmie BFS.

2.6. Algorytmy znajdowania ścieżki

PathfindingStrategy
Interfejs dla strategii znajdowania najkrótszej ścieżki. Zawiera metody `decideMove()` oraz `findFastestPath()`, która wyznacza optymalną trasę.

RandomFastestPathStrategy
Losowy algorytm znajdowania ścieżki.

DijkstraPathfindingStrategy
Strategia znajdowania najkrótszej ścieżki oparta na algorytmie Dijkstry.

2.7. Narzędzia pomocnicze

Randomizer
Klasa umożliwiająca generowanie pseudolosowych wartości i kierunków ruchu.