

Zaawansowane C++

Wstępna analiza funkcjonalna projektu

Wybrany temat: **Micromouse**

Autor: Jan Augustyn, 342201

1. Opis działania programu

Najważniejszą częścią programu będzie symulacja ruchów myszy podejmowanych na podstawie czujników wykrywających pobliskie ściany oraz ich wizualizacja w interfejsie użytkownika. Program powinien wyświetlać wygenerowany labirynt, mysz skierowaną w odpowiednim kierunku oraz elementy UI, pozwalające użytkownikowi na konfigurację oraz kontrolowanie przebiegu symulacji.

W pierwszej fazie, użytkownik powinien wygenerować labirynt o zadanych parametrach. Następnie może on wybrać typ myszy, czyli algorytm wykorzystywany do eksploracji (mapowania) labiryntu i metodę wyboru optymalnej ścieżki. Ostatecznie użytkownik powinien móc uruchomić symulację w dwóch fazach: eksploracji oraz rozwiązywania. W pierwszej fazie, robot będzie poruszał się po mapie w celu rozpoznania i zapamiętania układu labiryntu. Natomiast w fazie drugiej, robot, wykorzystując nabytą w fazie eksploracji wiedzę, powinien przemieścić się z miejsca startowego do końcowego w możliwie najszybszy sposób.

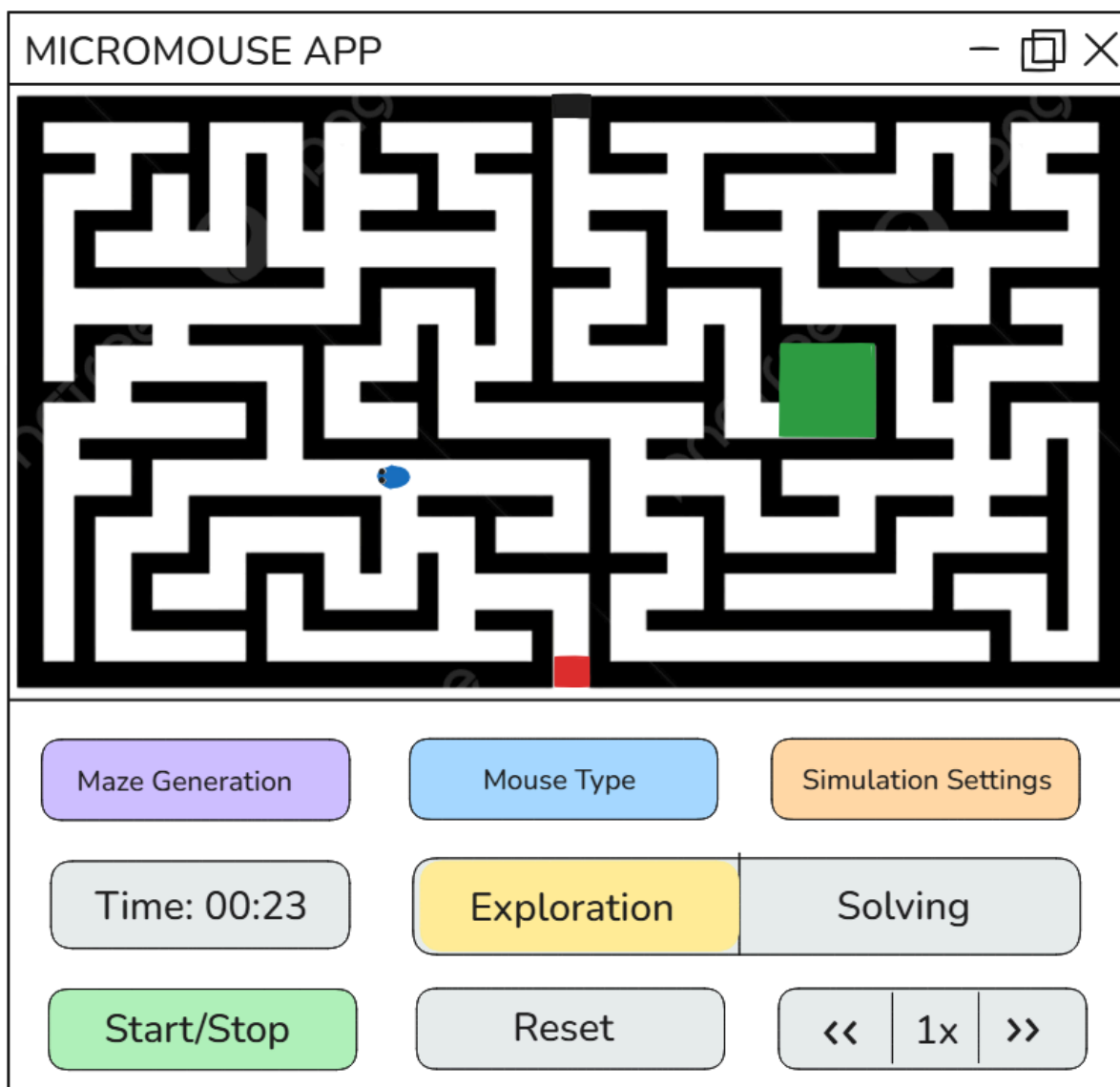
W wersji podstawowej, program powinien umożliwić generowanie labiryntu o zadanym przez użytkownika rozmiarze. Dostępne powinny być co najmniej dwa różne rodzaje algorytmów eksploracji i optymalizacji ścieżki. Z poziomu UI użytkownik będzie mógł uruchomić, zatrzymać, wznowić i zresetować daną symulację oraz przełączać pomiędzy fazami eksploracji i rozwiązywania.

Rozszerzeniami mogłyby być np. wprowadzenie większej liczby parametrów generowania labiryntu, większej liczby dostępnych algorytmów lub wzbogacenie interfejsu użytkownika o możliwość przyśpieszenia/spowolnienia liczby kroków symulacji w czasie.

2. Wymagania funkcjonalne

- Mysz podejmuje decyzje tylko w oparciu o lokalne dane.
- Generowanie labiryntu o zadanym rozmiarze.
- Symulacja kolejnych kroków symulacji oraz ich wizualizacja.
- Możliwość kontroli oraz konfiguracji symulacji z pozycji interfejsu użytkownika.
- Możliwość wyboru algorytmu eksploracji i optymalizacji ścieżki.

3. Szkic interfejsu użytkownika



- “Maze Generation” – wybór rozmiaru labiryntu i opcjonalnie poziomu skomplikowania.
- “Mouse Type” – wybór algorytmu eksploracji i optymalizacji.
- “Simulation Settings” – ewentualne dodatkowe opcje symulacji.
- “Time” - czas liczony od rozpoczęcia danej fazy symulacji.
- “Exploration/Solving” - przełącznik pomiędzy trybem eksploracji i optymalnego przejścia myszy.
- “Start/Stop” - Włączenie/zatrzymanie symulacji.
- “Reset” - Reset aktualnej fazy (powrót robota na miejsce startowe oraz wyzerowanie czasu).
- “<< 1x >>” - sterowanie prędkością symulacji.

Przyciski “Maze Generation”, “Mouse Type” i “Simulation Settings” w zamyśle powinny otworzyć oddzielne okno z odpowiednimi opcjami.

4. Wybrane narzędzia

- Środowisko: JetBrains CLion lub Visual Studio 2022 Community Edition
- Język: C++20
- Kompilator: GCC 9+ lub MSVC
- Biblioteki GUI: SFML + TGUI