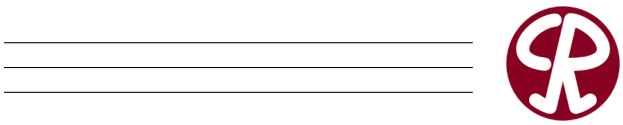
Projekt

Wizualizacja Danych Sensorycznych

Wezuwiusz

Katarzyna Czarnacka, 241517
Adam Królewiecki, 241480



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

9 kwietnia 2022

Spis treści

1 Opis

Zadaniem projektowym jest stworzenie wizualizacji dla ruchu myszy (obiekt 3D) po planszy, na której będą widoczne postępy w odszukiwaniu wyjścia z labiryntu. Dodatkowo, sama plansza ma być interaktywna, poprzez zewnętrzny moduł STM z żyroskopem i akcelerometrem. Przy pomocy tego modułu będzie można sterować lawą, poprzez zmianę nachylenia planszy. Celem osoby sterującej, jest dogonienie myszy. Sama mysz posiadać będzie akcelerometr i żyroskop, enkodery, diody i fotokomórki, dzięki którym będzie rejestrować otoczenie. Aby zagwarantować dostarczenie projektu, zostanie wykonana czysto symulacyjna wersja micromouse, która będzie wyjściem bezpieczeństwa, w razie niezrealizowania założeń projektu fizycznego. Praca zostanie podzielona na część związana z planszą i cześć związaną z robotem.

Ze względu na równoległą realizację projektu micromouse, w początkowej fazie projektu zostanie wykorzystana symulacja, która na dalszym etapie zostanie zastąpiona robotem.

1.1 Finalny efekt

Aplikacja będzie wizualizować trasę myszy, a za pomocą akcelerometru umieszczonego w telefonie, będzie można poruszać planszą o wymiarach $1.5m \times 1.5m$ (w celu kierowania lawą). Trasa myszy będzie rysowana w czasie rzeczywistym (z ewentualnym niewielkim opóźnieniem), podobnie będzie zachowywała się plansza z lawą.

1.2 Założenia projektowe

Aby aplikacja działała poprawnie, niezbędne będą poniższe kroki:

- poprawne zdefiniowanie wirtualnej lokalizacji robota,
- poprawne wyświetlanie modelu robota, planszy i kuli,
- przyjmowanie na bieżąco odczytów z sensorów zamontowanych na robocie,
- zrealizowanie komunikacji pomiędzy telefonem, a komputerem i programem,
- poprawna interpretacja ruchów planszy, sterowanej żyroskopem,
- utworzenia interpretatora danych sensorycznych.

1.3 Pod cele projektu

Bardziej szczegółowe przedstawienie zagadnień związanych z danym tematem. Wyodrębnienie pod celów.

Lista pod celów:

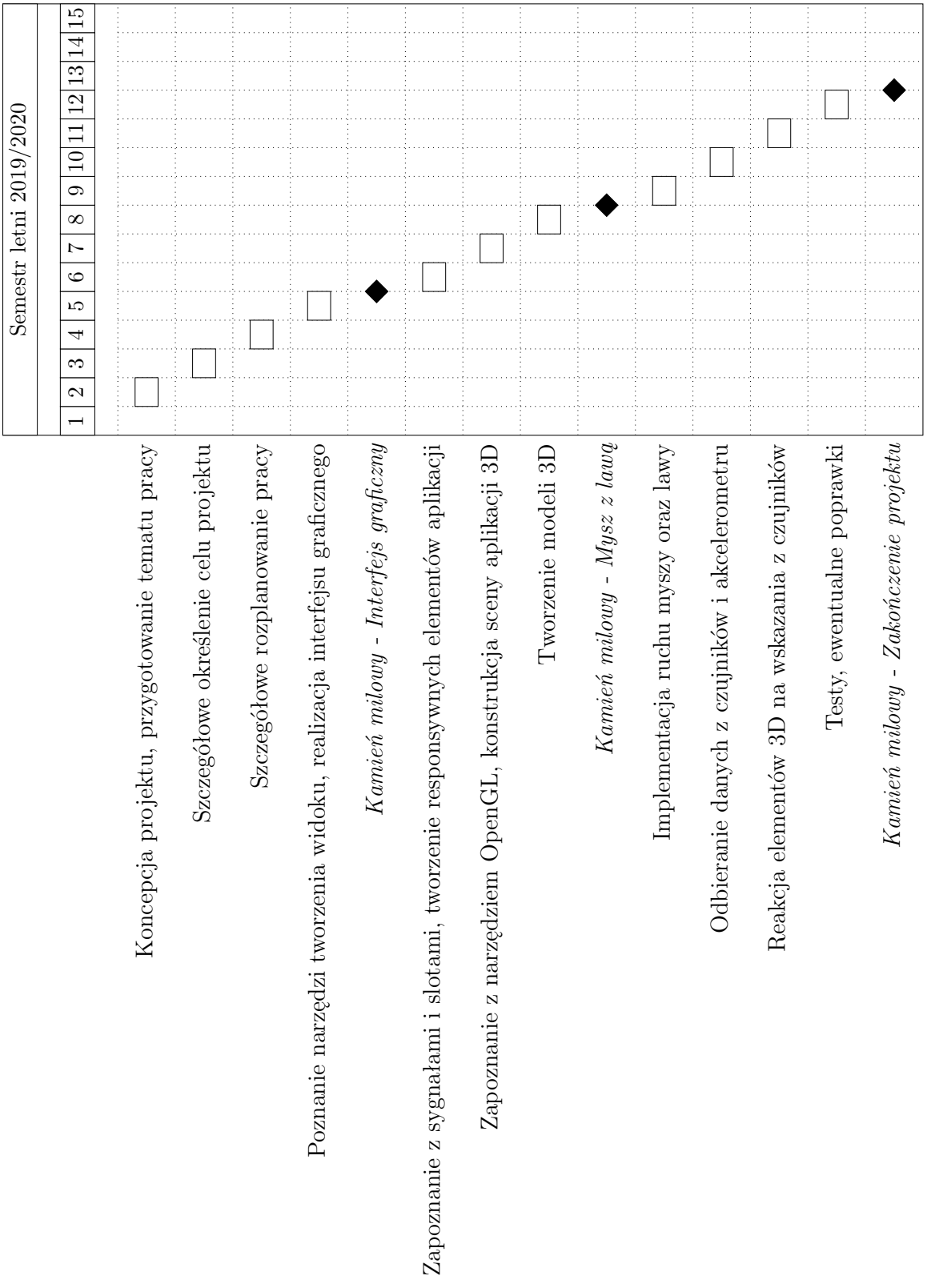
- przegląd literatury i zasobów Internetu związanych z tematem projektu,
- projekt układu elektronicznego (schemat ideowy),
- zapoznanie się z niezbędnymi funkcjami biblioteki Qt,

- uruchomienie programu do wizualizacji pojazdu na scenie,
- utworzenie oprogramowania odpowiedzialnego za komunikacje z komputerem,
- utworzenie oprogramowania odpowiedzialnego za analizę i interpretację danych sensorycznych.

2 Harmonogram

- Tydzień I 9-15.03.2020
  Koncepcja projektu, przygotowanie tematu pracy
- Tydzień II 16-22.03.2020
  Szczegółowe określenie celu projektu
- Tydzień III 23-29.03.2020
  Szczegółowe rozplanowanie pracy instalacja środowiska
- Tydzień IV 30-5.04.2020
  Poznanie narzędzi tworzenia widoku, realizacja okienka interfejsu graficznego
  KC i AK - interfejs graficzny z spersonalizowaną szatą graficzną
- Tydzień V 6-12.04.2020
  Zapoznanie z sygnałami i slotami, tworzenie responsywnego elementu aplikacji
  KC i AK - stworzenie paska postępu z spersonalizowaną szatą graficzną
- Tydzień VI 13-19.04.2020
  Zapoznanie z narzędziem OpenGL, konstrukcja sceny aplikacji 3D
  KC - konstrukcja sceny aplikacji 3D, AK - parametryzacja sceny
- Tydzień VII 20-26.04.2020
  Tworzenie modeli 3D
  KC - konstrukcja modelu lawy, AK - konstrukcja modelu myszy
- Tydzień VIII 27-3.05.2020
  Implementacja ruchu myszy oraz lawy
  KC - implementacja ruchu lawy, AK - implementacja ruchu myszy
- Tydzień IX 4-10.05.2020
  Odbieranie danych z czujników i akcelerometru
  KC - implementacja modułu danych z akcelerometru (telefon),
  AK - implementacja modułu odbierania danych z czujników robota
- Tydzień X 11-17.05.2020
  Reakcja elementów 3D na wskazania z czujników
  KC - fizyka modelu lawy, AK - fizyka modelu myszy
- Tydzień XI 18-24.05.2020
  Testy, ewentualne poprawki

2.1 Wykres Gantta



Rysunek 1: Diagram Gantta

2.2 Kamienie milowe

2.2.1 Interfejs graficzny

Wyświetlany interfejs graficzny, utworzenie interaktywnych przycisków i przykładowego modelu, wyświetlanego dla użytkownika.

2.2.2 Mysz z lawą

Statyczny model 3D myszy i lawy. Prawidłowy render planszy, robota i lawy, która ma za nim podążać. Nałożenie tekstur. Czytanie danych sensorycznych.

2.2.3 Zakończenie projektu

Sfinalizowany projekt z wprowadzonymi poprawkami. Poruszanie modelami w czasie rzeczywistym. Wykrywanie kolizji. Sprawny moduł komunikacyjny.

2.3 Podział pracy

Katarzyna Czarnacka	**Adam Królewiecki**
Moduł komunikacji z programem ruszającym planszą	Stworzenie interfejsu aplikacji
Wizualizacja planszy	Wizualizacja robota, implementacja testowego programu ruchu
Stworzenie modułów lawy	Algorytmy generujące plansze

Tabela 1: Podział pracy

3 Projekt graficznego interfejsu użytkownika

3.1 Funkcjonalności

Poniżej zostały opisane wspólne funkcjonalności przycisków zastosowanych w aplikacji. Z kolei w podsekcjach 3.1.1 oraz 3.1.2 zostały zawarte funkcjonalności przycisków, charakterystyczne dla danych okienek.

- Ekran sceny
  Główne okno będzie przedstawiać utworzone modele, ich interakcje i ruchy, które samo w sobie nie będzie interaktywne,
- Przycisk startu
  Przycisk odpowiedzialny za rozpoczęcie lub wznowienie symulacji, możliwości ruchu planszy, lawy i robota,
- Przycisk resetu
  Przycisk odpowiedzialny za zresetowanie symulacji,
- Wskaźnik czasu do uwolnienia lawy
  Timer, który pokazywać będzie pozostały czas przed pojawieniem na planszy lawy,

- Pole zmiany czasu uwolnienia lawy
  Pole, które pozwala na zadanie czasu uwolnienia lawy przez użytkownika (jednak musi się ona zawierać w zadanym przedziale od 0.0 do 60.0 sekund),

- Wskaźnik połączenia z zewnętrznym żyroskopem
  Wskaźnik, pokazujący czy połączenie z zewnętrznym żyroskopem, zielony kolor oznacza połączenie, czerwony brak połączenia,

- Stoper
  Timer pokazujący czas w jakim myszy znalazła wyjście z labiryntu, bądź też dogonienie jej przez lawę,

- Checkbox zmiany widoku
  Zaznazenie powoduje zmianę widoku kamery na rzut z góry na planszę,

- Checkbox śledzenia myszy
  Zaznaczenie pozwala na obserwację ruchu myszy, dzięki czemu niezależnie od położenia myszy, będziemy ją widzieć, nawet gdy będzie znajdowała się za ścianą (efekt ten zostanie uzyskany przez wprowadzenie częściowej przezroczystości ścian labiryntu w miejscu gdzie znajduje się mysz),

- Checkbox trasy myszy
  Zaznaczenie pozwala na pozostawianie śladu myszy w labiryncie

3.1.1 Symulacja

- Przycisk stopu
  Przycisk odpowiedzialny za zatrzymanie symulacji, spauzowany zostaje ruch planszy, lawy i myszy,

- Przyciski zmiany prędkości symulacji myszy
  Przycisk odpowiedzialny za zwiększenie lub zmniejszenie prędkości, z jaką symulowana mysz będzie się poruszać,

- Lista algorytmów
  Lista wyboru, pozwalająca wybrać stosowany w symulacji algorytm ruchu myszy z kilku dostępnych wariantów,

- Przycisk wizualizacji ruchu myszy
  Przycisk pozwala na przejście do trybu wizualizacji ruchu myszy,

3.1.2 Ruch myszy

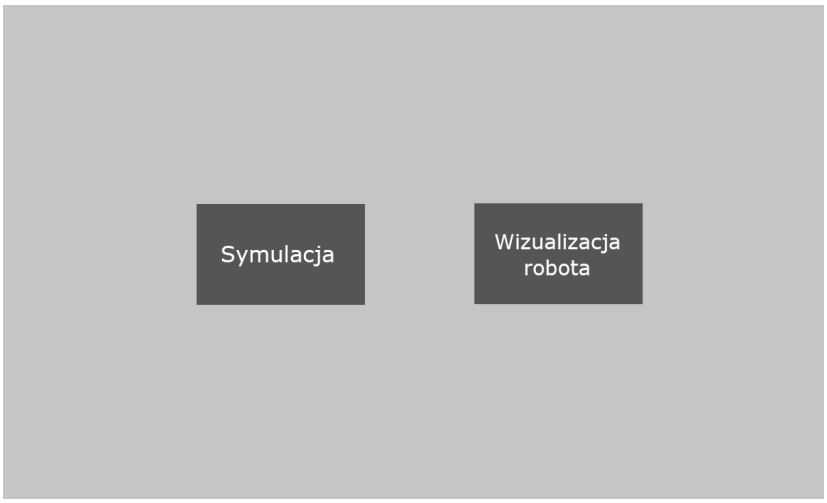
- Przycisk stopu
  Przycisk odpowiedzialny za zatrzymanie części widoku, spauzowany zostaje ruch planszy i lawy, jednak mysz porusza się nadal

- Wskaźnik połączenia z robotem
  Wskaźnik, pokazujący czy połączenie z robotem jest aktywne, zielony kolor oznacza połączenie, czerwony brak połączenia

- Przycisk wizualizacji symulacji
  Przycisk pozwala na przejście do trybu wizualizacji symulacji myszy dla algorytmów,

3.2 Wizualizacja

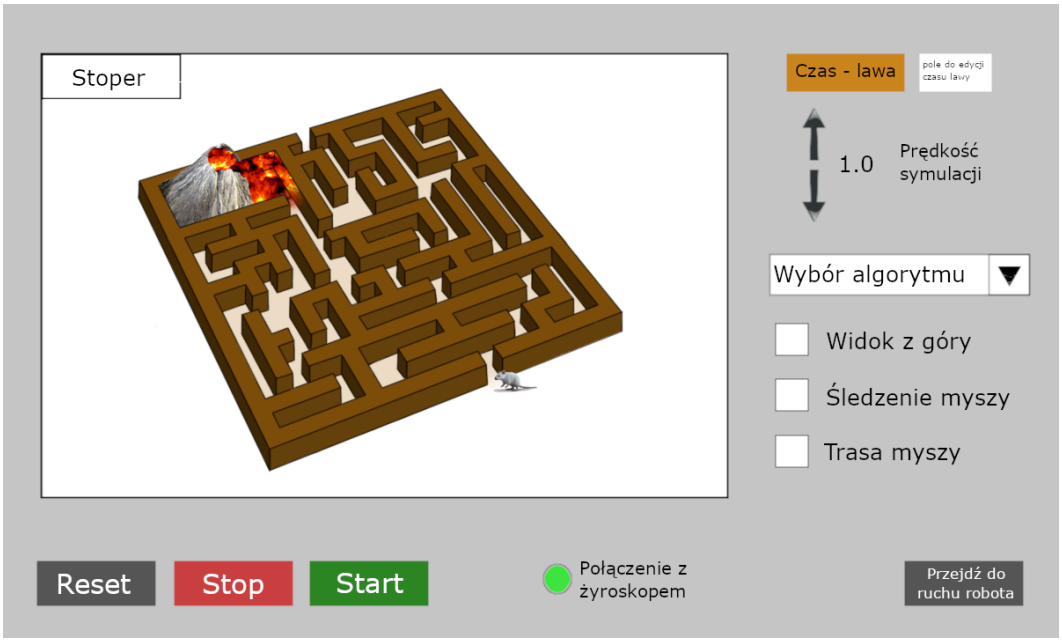
Po otworzeniu programu ukazuje się okno wyboru, które pozwala określić tryb wizualizacji. Możliwy będzie wybór symulacji trasy myszy obliczonej za pomocą algorytmu lub wizualizacji ruchu robota. Okno wyboru 2 będzie wielkości docelowego okna symulacji. Po wyborze nadal będzie możliwość zmiany symulacji jednak już za pomocą przycisku w oknie symulacji.



Rysunek 2: Okno wyboru

3.2.1 Okno symulacji

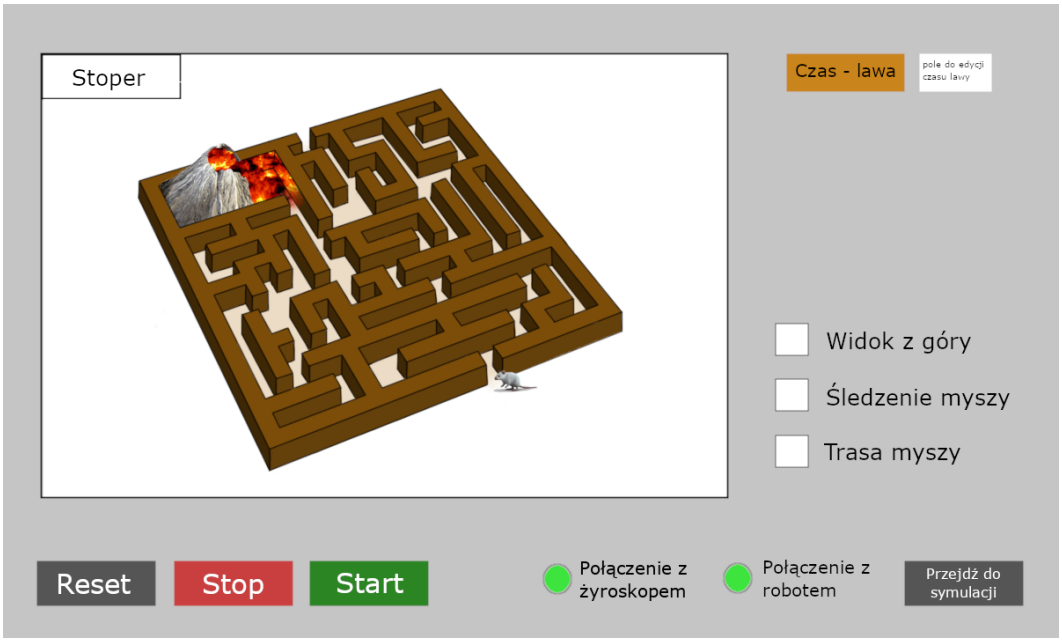
Po wyborze SYMULACJA w oknie z rysunku 2 wyświetla się okno przedstawione na rysunku 3. Domyślnie po wyświetleniu okna zostanie wybrany pierwszy algorytm.



Rysunek 3: Okno symulacji

3.2.2 Okno ruchu myszy

Po wyborze Wizualizacja robota w oknie z rysunku 2, wyświetla się okno z przedstawione na rysunku 4.



Rysunek 4: Okno symulacji

3.3 Scenariusze działania aplikacji

3.3.1 Tryb symulacji

Po wyborze Symulacja w oknie z rysunku 2, zostaje wybrany domyślny algorytm, a cała symulacja jest w trybie Stop. Przed jej rozpoczęciem użytkownik może ustawić spersonalizowane parametry, takie jak czas uwolnienia lawy, widok, trasę myszy, algorytm bądź prędkość symulacji. Przyjęte ustawienia będą mogły zostać częściowo zmienione po za pauzowaniu symulacji przyciskiem Stop. Symulacja może zostać zresetowana przy pomocy przycisku Reset, kiedy to będzie zależy od użytkownika. Po przejściu przez mysz planszy, lawa będzie nadal aktywna.

3.3.2 Tryb śledzenia ruchu robota

Po wyborze Wizualizacja robota w oknie z rysunku 2, następuje przejście do symulacji, która jest w trybie Stop. Przed jej rozpoczęciem użytkownik może ustawić czas uwolnienia lawy, widok bądź trasę myszy. Przycisk Stop, pozwala na spauzowanie jedynie ruchu lawy, a mysz nadal się porusza. Symulacja może zostać zresetowana przy pomocy przycisku Reset, jednak wymaga to ustawienia robota na początku labiryntu. Po przejściu przez mysz planszy, widok na labirynt zostaje zamrożony, a aplikacja przestaje przyjmować dane z robota. Żeby rozpocząć od nowa całość należy nacisnąć Reset, a po wprowadzeniu odpowiednich ustawień Start.

4 Realizacja graficznego projektu użytkownika

4.1 Funkcjonalność

4.1.1 Symulacja

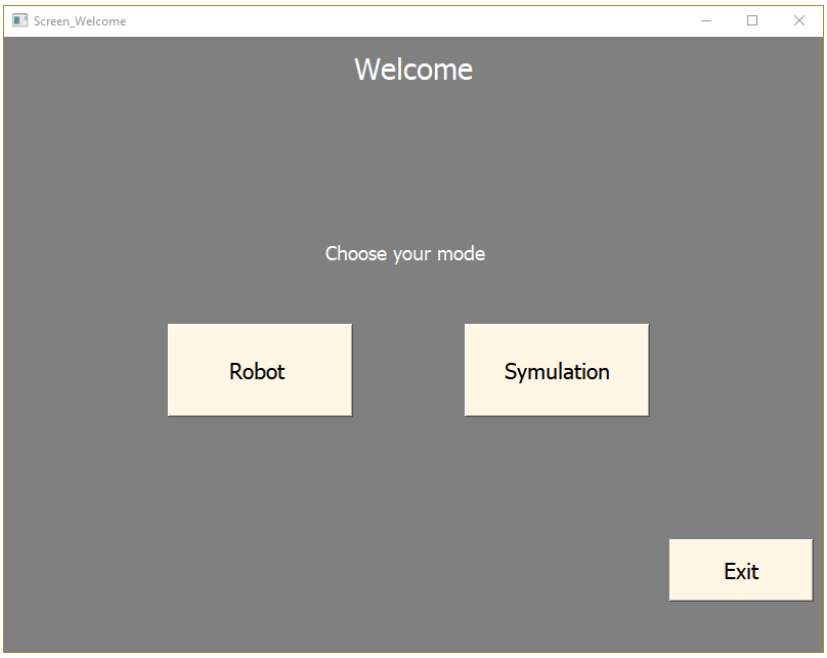
Zgodnie z opisem projektu z punktu 3.1.1, wraz wyborem widoku symulacji, zostaje wysłany sygnał, który tymczasowo zmienia wartość zadanej etykiety. W dalszej realizacji projektu zostanie zaimplementowane okno OpenGL, które umożliwi wyświetlenie mapy 3D. Analogicznie działa to w trybu śledzenia myszy oraz jej trasy. Powyżej wyboru algorytmu został zawarty timer erupcji lawy. Zaimplementowane wskaźniki w zależności od otrzymanego sygnału zmieniają barwę. Stoper odlicza czas od startu symulacji.

4.1.2 Śledzenie ruchu robota

Zgodnie z opisem projektu z punktu 3.1.2, wraz wyborem widoku symulacji, zostaje wysłany sygnał, który działa analogicznie jak w trybie symulacji. Powyżej wyboru algorytmu został zawarty timer erupcji lawy. Stoper odlicza czas od startu ruchu robota. Działanie wskaźnika połączenia działa analogicznie jak w trybie symulacji.

4.2 Realizacja aplikacji

Po otworzeniu aplikacji jest wyświetlany ekran wyboru, przedstawiony na rysunku. 5. Pozwala on na wybór trybu pracy programu, jak i opuszczenie go.



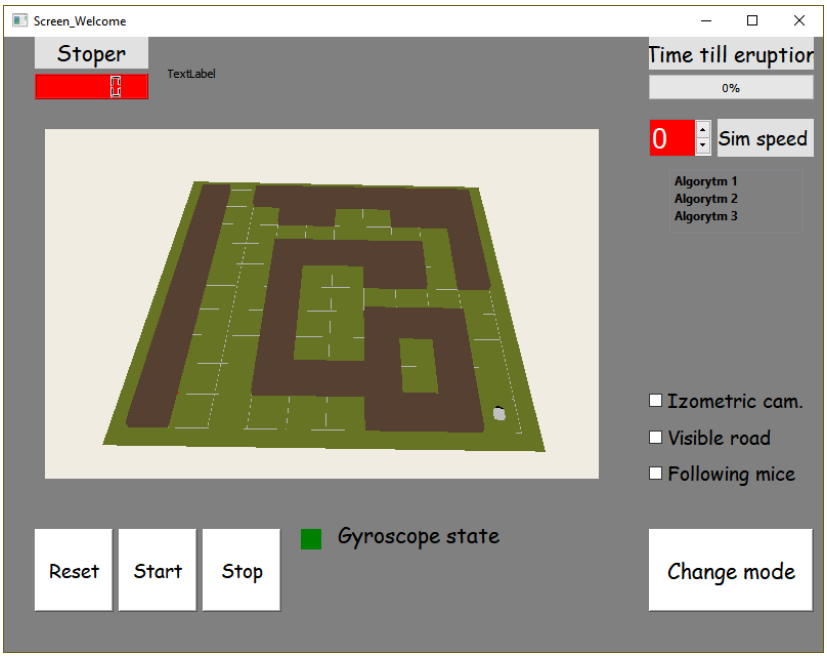
Rysunek 5: Okno wyboru

4.2.1 Okno symulacji

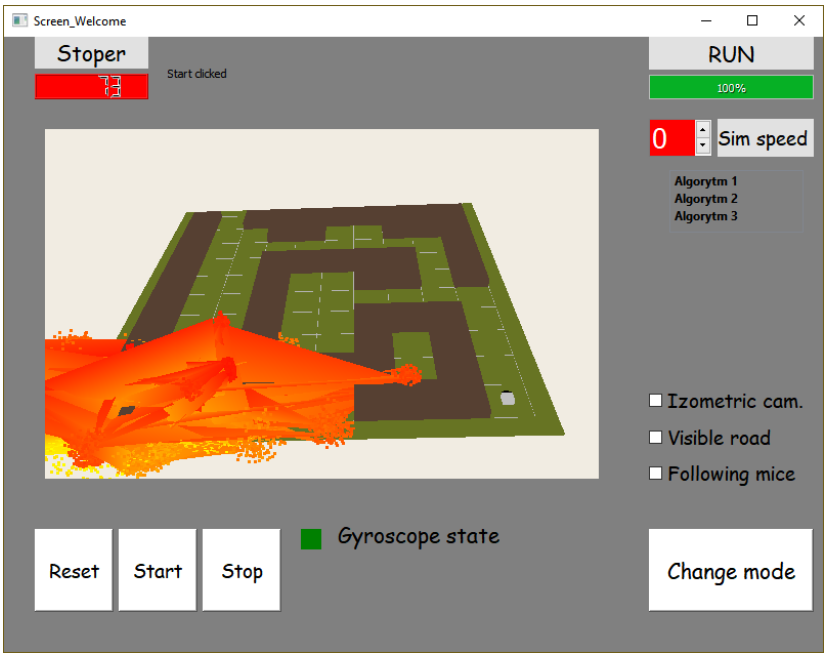
Po wyborze SYMULATION w oknie 5 wyświetla się okno przedstawione na rysunku 6. Domyślnie po wyświetleniu okna jest wybrany pierwszy algorytm. Po naciśnięciu przycisku START rozpoczyna się odliczanie do erupcji, a następnie następuje wyrzut

lawy, która jest interaktywna i reaguje na zmianę wartości odczytu akcelerometru z telefonu.

Działanie symulacji zostało przedstawione na rysunku 7. Niestety nie zostały one całkowicie zaimplementowany. Labirynt jest tworzony na podstawie tablicy znaków oraz polygonów. Obraz myszy generowany jest na podstawie punktu pozycji, który pozwala na automatyczne wygenerowanie całego modelu. Dostępne są również poboczne funkcjonalności poza wyborem algorytmu oraz trybami kamery.



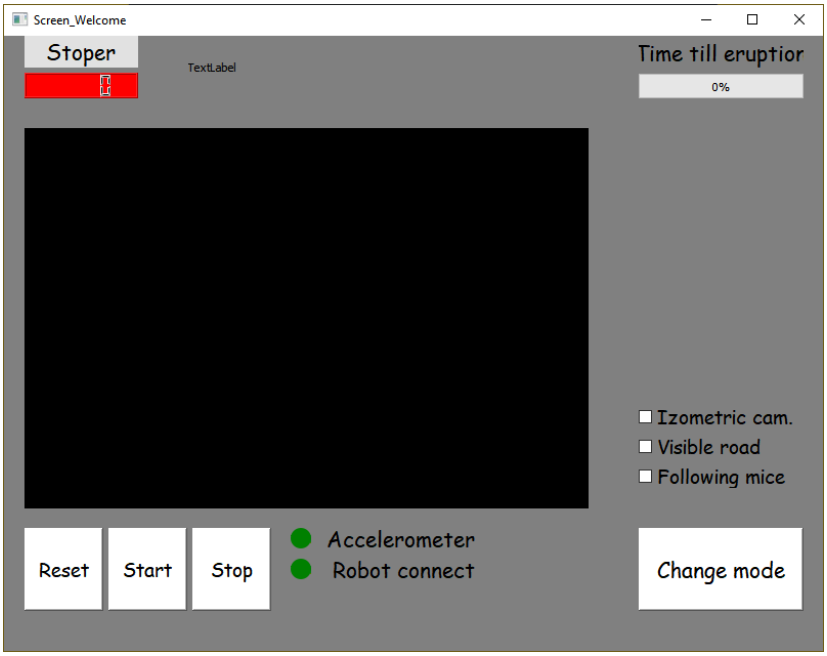
Rysunek 6: Okno symulacji



Rysunek 7: Start symulacji

4.2.2 Okno ruchu myszy

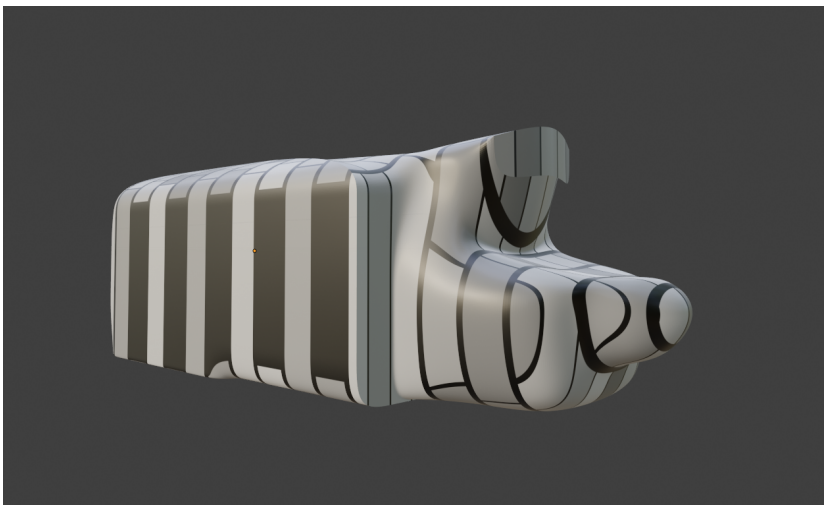
Niestety nie udało się zrealizować połączenia z symulacji z robotem. Po wyborze w oknie z rysunku 5 przycisku ROBOT, następuje przejście do nieaktywnego okna trybu śledzenia robota, widok okna został przedstawiony na rysunku 8. Program zostałby zaimplementowany analogicznie jak w przypadku symulacji. Jednak różniłby się tym, że ruch robot nie byłby rysowany na podstawie wczytanego algorytmu, tylko z odczytu czujników (enkoderów oraz żyroskopu).



Rysunek 8: Okno w trybie śledzenia ruchu robota

5 Model Myszy

Został także stworzony model myszy 3D, przedstawiony na rysunku 9, który w założeniach miał zostać zaimplementowany w programie. Jednak ze względu na trudności związane z konwersją modelu .obj do .mesh, nie został on przeniesiony do symulacji. Finalnie model myszy został narysowany na podstawie polygownów za pomocą funkcji oferowanych przez *Qgl Viewer*. Dodatkowo wykorzystano także klasy i metody oferowane przez bibliotekę QT [2], takie jak wektory czy



Rysunek 9: Model myszy

6 Symulacja

Symulacja została zrealizowana wraz z użyciem elementów takich jak

- Wulkan
- Mysz
- Labirynt
- Plansza

Zgodnie z założeniami całość realizowana jest w grafice 3D, rysowana w *QGL Viewer* [1] przy pomocy obiektów Polygon. Proces ten został zautomatyzowany za pomocą algorytmów, które na bazie tablicy znaków char generują wybrany kształt labiryntu.

Mysz jest w stanie obracać się wokół własnej osi oraz poruszać się w losowe kierunki, które są dla niej dostępne. Bardziej skomplikowane algorytmy zostały użyte w fizycznym odpowiedniku gryzonia.

Wulkan jest responsywny na ruchy akcelerometru w telefonie, co prowadzi do spontanicznych erupcji lawy.

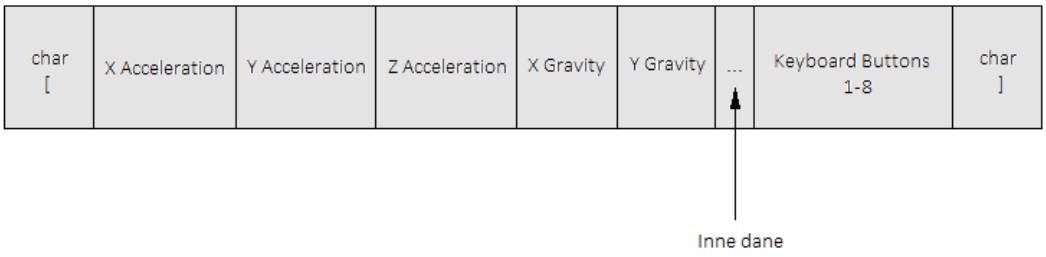
7 Realizacja połączeń

7.1 Protokół UDP

Protokół TPC/IP został zamieniony na protokół UDP ze względu na zagwarantowanie przez drugi łatwiejszego odbioru danych, w których nie trzeba potwierdzać otrzymania pakietu danych. Drugim powodem zmiany decyzji początkowej projektu było wycofanie aplikacji z dystrybucji.

Dane z telefonu są przesyłane za pomocą protokołu UPD, który umożliwia transmisję danych między programem, a urządzeniem. Zaimplementowane zostało rozróżnienie klientów dzięki któremu możliwe będzie odbieranie danych od odpowiedniego klienta (akcelerometru). Za pomocą telefonu wysyłany jest pakiet danych w postaci trzech

współrzędnych oddzielonych od siebie znakiem w postaci przecinka, kolejno współrzędne to xyz, pozostałe dane, to odczyty z żyroskopu, kąta RPY itp. jednak nie są one przesyłane do aplikacji. Wykonany został również diagram przesyłu danych, przedstawiony na rysunku 10. W celu implementacji protokołu wykorzystano wiedzę w zakeresie użycia protokołu UDP [4] oraz wiedzę na tmat wątków [3].



Rysunek 10: Ramka danych odbieranych z akcelerometru

8 Podsumowanie

Udało się zaimplementować animacje oraz połączyć ją z interfejsem aplikacji, a także zrealizować rysowanie labiryntu i ruch lawy. Większość założeń została spełniona, a pozostałe z nich, będą mógłby być realizowane w postaci hobbystycznej, po zaprogramowaniu robota będzie można odzwierciedlić jego ruch w programie. Jednym z możliwych rozwojów aplikacji jest także poprawna implementacja algorytmów czy udoskonalanie fizyki bądź implementacja ruchu kamery.

Literatura

[1] Frédéric Boudon. libqglviewer. http://libqglviewer.com/.

[2] Haavard Nord, Eirik Chambe-Eng. Qt documentation. https://doc.qt.io/.

[3] VoidRealms. C++ qt 28 - qthread part 1 creating a thread. https://www.youtube.com/watch?v=JaGqGhRW5Ks.

[4] VoidRealms. C++ qt 71 qudpsocket. https://www.youtube.com/watch?v=4qx4FaglSig.