

Wizualizacja danych sonaru

Dorian Janiak

14.05.2015

1 Krótki opis

Projekt zakłada stworzenie aplikacji komputerowej, która będzie wizualizowała otrzymywane dane z sonaru ultradźwiękowego. Aplikacja ma próbować łączyć dane w taki sposób, aby móc z nich stworzyć zarys otoczenia. Drugą część projektu stanowi symulacja, której zadaniem jest z odpowiednio przygotowanych obiektów geometrycznych wygenerować dane potrzebne do wykonania wizualizacji (ma być symulacją sonaru). Dane mają zostać przesłane przy pomocy wybranego protokołu komunikacyjnego.

2 Cel

Celem projektu jest zapoznanie się ze środowiskiem Qt oraz sposobami wizualizacji danych. Jednym z problemów, który zostanie również poruszony w ramach pracy będzie sposób komunikacji modułów i przesył danych. Jednym z ważnych powodów podjęcia się realizacji tego tematu jest chęć głębszego poznania biblioteki OpenGL.

3 Rozszerzony opis

3.1 Komputer PC

Projekt składa się z dwóch części. Pierwsza część obejmuje stworzenie aplikacji w Qt, która pozwoli na przedstawienie danych sensorycznych z sonaru. Sonar i robot mobilny zostaną stworzone w ramach osobnego projektu (kurs: Roboty mobilne(1)). Aplikacja stworzona w środowisku Qt pozwala również na sterowanie robotem. Dane są wysyłane do robota przy użyciu interfejsu Bluetooth. Robot reaguje na to zmieniając swoją pozycję. Następnie z aplikacji Qt wysyła się żądanie wykonania skanowania terenu. Robot przy pomocy czujnika ultradźwiękowego rozpoczyna skanowanie terenu, zapisując dane w swojej pamięci. Gdy skończy - przesyła dane pomiarowe przez interfejs Bluetooth do komputera z aplikacją. Dane zostają wczytane i zwizualizowane. Należy przy tym pamiętać, że robot nie znajduje się dokładnie w tym miejscu, w którym oczekivalibyśmy. Dane zostają połączone przez aplikację i wyrysowany zostaje kolejny kawałek mapy otoczenia. Gdyby omiatanie terenu odbywało się również wertykalnie, a nie tylko horyzontalnie można by wymagać większej dokładności wizualizacji.

3.2 Urządzenie

Drugą część aplikacji stanowi symulator robota z sonarem. Początkowym założeniem było stworzenie symulatora uruchamianego na tym samym komputerze co główna aplikacja, jednak po rozpoznaniu tematu i ponownym oszacowaniu dostępnego czasu projekt symulatora został ograniczony do projektu aplikacji na urządzenie z systemem Android. Aplikacja będzie wczytywać obiekty z plików OBJ oraz ze specjalnych plików konfiguracyjnych położenie poszczególnych obiektów na scenie (proponowany format: JSON). Następnie będzie omiatła otoczenie z pewną dokładnością. Wyniki zostaną

wysłane poprzez interfejs Bluetooth. Posłużą one do zwizualizowania efektów symulacji w oknie aplikacji z pierwszej części. Ostatnim etapem rozwoju symulatora będzie stworzenie ciągłej komunikacji między symulatorem i aplikacją wizualizującą, tak aby można było sterować robotem umieszczonym w symulatorze.

4 Funkcjonalność skończona

Poniżej zamieszczona została lista funkcjonalności, którą udało się ukończyć dotychczas. Lista zawiera również nieplanowany wcześniej punkt. Został on zrealizowany w celu ułatwienia późniejszego debugowania i wyszukiwania problemów w trakcie komunikacji z urządzeniem. Klasa (MessageController) została specjalnie stworzona w taki sposób, aby dziedzicząc ją była w stanie obsłużyć zarówno dane pochodzące z pliku dyskowego jak i dane pochodzące z transmisji Bluetooth. *W nawiasach zostały zamieszczone dodatkowe uwagi odnośnie punktowanych funkcjonalności*

4.1 Aplikacja na komputerze PC

- Rysowanie na scenie 3D mapy otoczenia.
- Wczytywanie danych pomiarowych sonaru z pliku (w przypadku symulacji).
- Łączenie nowo odczytanej z poprzednio odczytaną mapą terenu.
- Możliwość sterowania widokiem 3D.
- *(Nieplanowane wcześniej)* Wyświetlanie logów dotyczących przesyłanych komunikatów oraz interpretacja występujących błędów.
- *(Zrealizowane w okresie między 1. i 2. raportem)* Rysowanie na scenie 3D robota.
- *(Zrealizowane w okresie między 1. i 2. raportem)* Udostępnienie pilota pozwalającego na sterowanie robotem (lub symulacją).

4.2 Symulator na urządzeniu

- *(Zrealizowane w okresie między 1. i 2. raportem)* Ładowanie plików OBJ, parsowanie oraz logowanie wczytanych informacji

5 Funkcjonalność planowana

Poniżej zamieszczone zostały listy funkcjonalności, które zamierzam wykonać w ramach projektu.

5.1 Aplikacja na komputerze PC

- Wysyłanie żądania przemieszczenia robota (poprzez interfejs komunikacyjny).
- Wysyłanie żądania wykonania skanowania terenu (poprzez interfejs komunikacyjny).
- Odbieranie danych z sonaru robota (poprzez interfejs komunikacyjny).

5.2 Symulator na urządzeniu

- Prosta symulacja sonaru ultradźwiękowego
- Wysyłanie danych do aplikacji wizualizującej poprzez interfejs komunikacyjny (Bluetooth)
- Odbieranie żądania przemieszczenia robota na scenie
- Odbieranie żądania wykonania symulacji i przesłania wyników

6 Funkcjonalność zawieszona

Funkcjonalność wymieniona w tym punkcie została zawieszona ze względu na ograniczenia czasowe oraz stosunkowo małą jej przydatność w programie. Symulator stworzony na komputerze PC przy poniższych założeniach nie jest w stanie dorównać symulatorowi zainstalowanemu na urządzeniu zewnętrznym, z którym program będzie musiał komunikować się poprzez interfejs Bluetooth. Czas, który pierwotnie miał być poświęcony na jej stworzenie zostanie przeznaczony na stworzenie symulatora na telefonie z systemem Android.

6.1 Symulator na komputerze PC

- Ładowanie plików konfiguracyjnych (zawierają informacje o położeniu obiektów 3D na symulowanej scenie)
- Ładowanie plików OBJ
- Symulacja sonaru ultradźwiękowego
- Zapisanie danych do z góry ustalonego pliku wynikowego (np. out.txt)

6.2 Symulator na urządzeniu

- *(Zawieszone w okresie między 1. i 2. raportem)* Ładowanie plików konfiguracyjnych (zawierają informacje o położeniu obiektów 3D na symulowanej scenie)

7 Lista kamieni milowych

- K1 - zakłada działającą podstawową aplikację komputerową. Aplikacja jest w stanie rysować wszystkie potrzebne do symulacji obiekty 3D. Udostępnia również prosty pilot. Jest w stanie załadować prosty plik symulacyjny. Pozwala na sterowanie widokiem 3D.
- K2 - zakłada podstawową formę komunikacji między urządzeniem opartym o system Android, a aplikacją komputerową. Nie zakłada pełnej interakcji, ale urządzenie jest w stanie przesyłać takie podstawowe informacje jak pozycja robota oraz wynik pomiaru danych. Niekoniecznie obsługiwana jest jeszcze pozycja zadawana przez aplikację komputerową.
- K3 - zakłada stworzone: aplikację oraz symulator. Obie są w stanie się ze sobą komunikować oraz dodatkowo aplikacja komputerowa komunikuje się z robotem.

8 Harmonogram

Poniższa tabela zawiera harmonogram przyjęty jako obowiązujący w okresie od oddania raportu nr 1. Został on zaktualizowany o informacje, które zadania zostały ukończone.

Od:	Do:	Harmonogram tworzenia głównej aplikacji	Harmonogram tworzenia symulatora	Kamień mi- lowy	Status
23.03	29.03	Stworzenie szkieletu aplikacji QT z ładowaniem okna 3D.			Skończone
30.03	05.04	Przygotowanie diagramu klas oraz diagramu przypadków użycia.			Skończone
06.04	12.04	Rysowanie siatki, możliwość poruszania widokiem 3D. Parsowanie pliku symulacyjnego.			Skończone
13.04	19.04	Rysowanie mapy otoczenia oraz obsługa błędów i formatu wiadomości. Wyświetlanie logów komunikacji.			Skończone
20.04	26.04	Rysowanie robota w oknie 3D oraz obsługa zmiany jego położenia.		K1	Skończone
27.04	03.05		Zapoznanie z Android SDK.		Skończone
04.05	10.05	Stworzenie komunikacji przez Bluetooth. Parowanie urządzeń.	Stworzenie prostej komunikacji przez Bluetooth. Parowanie urządzeń.		Nierozpoczęte
11.05	17.05	Obsługa przesyłanych wiadomości poprzez Bluetooth.	Stworzenie symulatora sonaru. Ładowanie plików OBJ.	K2	Częściowo skończone
18.05	24.05	Obsługa przesyłanych wiadomości (z telefonu oraz robota)	Dopracowanie symulatora zgodnie ze sposobem działania robota.		
25.05	31.05	Poprawki obsługi modułu Bluetooth oraz synchronizacji widoku.	Dopracowanie symulatora zgodnie ze sposobem działania robota.		
01.06	07.06	Dostosowanie aplikacji do możliwości robota.			
08.06	14.06	Stosowanie poprawek.	Stosowanie poprawek	K3	

9 Zaktualizowany harmonogram

Poniższa tabela zawiera zmodyfikowany harmonogram, który będzie obowiązywał od aktualnego raportu do końca projektu.

Od:	Do:	Harmonogram tworzenia głównej aplikacji	Harmonogram tworzenia symulatora	Kamień mi- lowy	Status
23.03	29.03	Stworzenie szkieletu aplikacji QT z ładowaniem okna 3D.			Skończone

30.03	05.04	Przygotowanie diagramu klas oraz diagramu przypadków użycia.			Skończone
06.04	12.04	Rysowanie siatki, możliwość poruszania widokiem 3D. Parsowanie pliku symulacyjnego.			Skończone
13.04	19.04	Rysowanie mapy otoczenia oraz obsługa błędów i formatu wiadomości. Wyświetlanie logów komunikacji.			Skończone
20.04	26.04	Rysowanie robota w oknie 3D oraz obsługa zmiany jego położenia.		K1	Skończone
27.04	03.05		Zapoznanie z Android SDK.		Skończone
04.05	10.05		Stworzenie prostej aplikacji Android		Skończone
11.05	17.05		Stworzenie symulatora sonaru. Ładowanie plików OBJ.		Częściowo skończone
18.05	24.05	Stworzenie modułu komunikacji Bluetooth. Parowanie urządzeń.	Stworzenie obsługi komunikacji Bluetooth. Parowanie urządzeń.	K2	
25.05	31.05	Obsługa przesyłanych wiadomości (z telefonu oraz robota)	Dopracowanie symulatora zgodnie ze sposobem działania robota.		
01.06	07.06	Poprawki obsługi modułu Bluetooth oraz synchronizacji widoku.	Dopracowanie symulatora zgodnie ze sposobem działania robota.		
08.06	14.06	Stosowanie poprawek.	Stosowanie poprawek	K3	

10 Wygląd aplikacji

Poniżej prezentuję aktualnie uzyskany wygląd głównej aplikacji. Od czasu ostatniego raportu został on wyposażony w przyciski w panelu Kontrola, pozwalające na sterowanie obiektem, symbolizującym robota. W górnej części okna znajduje się menu, w którym znajdują się między innymi takie zakładki jak:

- "Plik- pozwoli na ładowanie z pliku symulacji oraz w razie decyzji dalszego rozwoju aplikacji na zapis i otwieranie różnego formatu plików
- "Bluetooth- pozwoli na parowanie urządzenia z aplikacją
- "Pomoc- będzie otwierało okno z informacjami o autorze.

Główną część okna stanowi widok 3D, w którym przy użyciu myszy komputerowej można sterować kątem kamery oraz jej przybliżeniem. W oknie tym rysowana jest mapa 3D. Na screenie widać poszczególne jej składowe (kolorowe linie) z tym, że wszystkie zostały wyrysowane względem tego samego punktu środkowego (0,0,0,1) we współrzędnych jednorodnych. Poniżej głównego okna znajduje

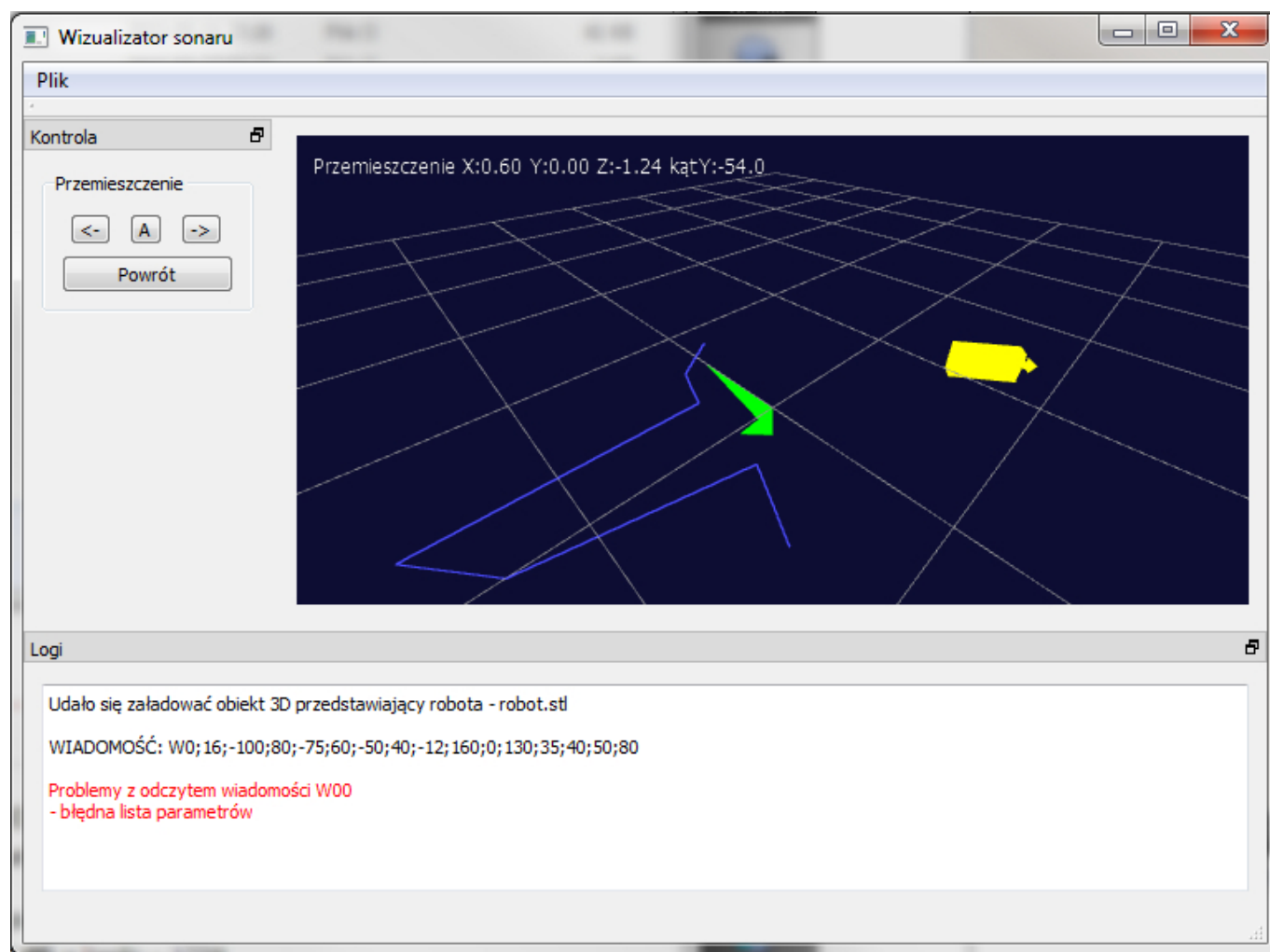
się dokowany widżet, w którym zapisywane są logi z aktualnie odbywającej się komunikacji. Zostały wyróżnione różnego typu komunikaty:

- kolor czarny - zwykła informacja
- kolor pomarańczowy - ostrzeżenie
- kolor czerwony - błąd (ale nie krytyczny)

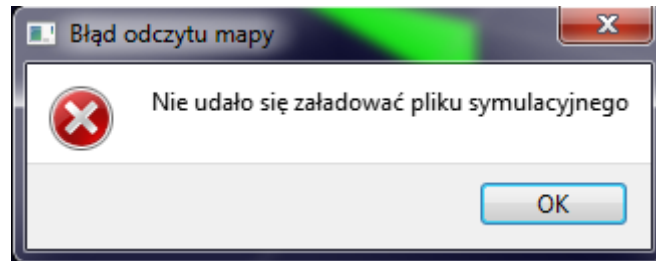
10.1 Zmiany w wyglądzie aplikacji

W czasie od złożenia ostatniego raportu (numer 1) zostały wprowadzone następujące zmiany:

- dodano panel w widżecie "Kontrola", który pozwala na sterowanie obiektem symbolizującym robota.
- dodano możliwość ładowania obiektu 3D zapisanego w formacie STL, który ma symbolizować robota. Ładowanie odbywa się przy starcie programu. Ładowany jest plik robot.stl. Informacja o statusie operacji zostaje ostatecznie zapisana w oknie logów.
- w oknie 3D pokazywane są informacje na temat przemieszczenia robota oraz jego orientacji wzdłuż osi Z (skierowana w górę).



Poważniejsze błędy, wymagające uwagi użytkownika są raportowane okienkiem błędów.



11 Diagramy

Poniżej zaprezentowany został diagram klas dla głównej aplikacji komputerowej. Analizę należy rozpocząć od klasy `MainWindow`, która odpowiada głównemu oknu aplikacji. Przechowuje ona obiekty pozostałych klas. `MessageController` odpowiada za interpretowanie i przygotowywanie wiadomości potrzebnych do komunikacji z urządzeniami oraz odczytem danych z plików. Będzie ona dziedziczona przez klasy `FileController` (obsługuje operacje plikowe) oraz `RobotController` (zarządza robotem). W przypadku tej drugiej klasy odziedziczy ona również po klasie `BluetoothController`. Zapewni to klasie `RobotController` komplet funkcji potrzebnych do sterowania robotem. Klasa `MapView` dziedziczy od klasy `QOpenGLWidget` oraz `QOpenGLFunctions` i odpowiada za sterowanie widokiem 3D. Klasa `EnvMap` przechowuje komplet informacji związanych z danymi skanowania. Przechowuje ona ją w postaci zbioru wierzchołków przestrzennych, które mogą zostać przekazane do kontekstu OpenGL, oraz dodatkowych zmiennych przechowujących informacje związane ze skalą obiektu, koloru materiału, punktu centralnego czy kąta obrotu.

W ramach projektu pozostało zaimplementowanie klas `BluetoothController` oraz częściowo już stworzonej klasy `RobotController`.

