Program do odkrywania świata z pomocą usługi sieciowej, wirtualny symulator czołgu

Autorzy:
Bartosz Śledzikowski, Bartłomiej Ignaciuk

Prowadzący:
Prof. Robert Szmurło

24 stycznia 2021

Zadania

1.1 Cel projektu

Celem projektu było stworzenie oprogramowania w języku C łączącego się z serwerem za po- mocą API i umożliwiającego sterowanie czołgiem. Zadanie zostało wykonane z wykorzystaniem bibliotek takich jak cJSON[1] (umożliwia łatwe parsowanie danych w formacie json w języku C) i curl[2] (odpowiada za komunikację z serwerem). Do generowania obrazu w formacie PNG została wykorzystana biblioteka libpng[4]. Ostatnim już z wykorzystanych zewnętrznych elementów był program indent[5]. Umożliwia on szybkie formatowanie napisanego kodu co znacznie poprawia jego czytelność.
Przykładowa komenda, którą możemy zrealizować taką operację:

```
indent -kr main.c
```

1.2 Kompilacja

Do kompilacji programu został napisany makefile. Wygląda on następująco:

```
main:
  gcc main.c apiConnector.c mapGenerator.c jsonConverter.c algorithm.c pngGenerator.c -Wall
                                                    -lcurl -lcjson -lpng16 -o tanks
test:
  gcc tests/mainTest.c tests/apiConnectorTest.c tests/mapGeneratorTest.c tests/
                                                    jsonConverterTest.c tests/algorithmTest.c -
                                                    Wall -lcurl -lcjson -lpng -o tanksTest

test-memory:
  valgrind ./tanks --leak-check=full clean:
  rm -f tanks
```

Wykorzystanie flagi -Wall pozwala na wykrywanie problemów w kompilowanym kodzie. Infor- muje również o problemach, które mogłyby zostać pominiętę przez ustawiony stopień optyma- lizacji.
Posiada on cztery opcje:

- main - kompiluje główny program.
- test - kompiluje program do testów.
- test-memory - uruchamia program główny w celu zbadania wycieków pamięci za pomocą narzedzia valgrind[3].
- clean - usuwa skompilowany program.

1.3 Moduły programu

Program składa się z różnych modułów. Możemy wyróżnić dwa główne:

- Główny - odpowiada za działanie docelowego programu.
- Testowy - pozwala na przeprowadzenie testów zaimplementowanych w głównym module metod.

W ramach gownego programu kod zostal podzielony na piec roznych modulow:

- main.c - moduł główny, odpowiada za wyświetlanie menu i uruchamianie wybranych przez użytkownika opcji.
- apiConnector.c - realizuje połączenie z serwerem.
- mapGenerator.c - przechowuje strukturę mapy i odpowiada za jej obsługę.
- jsonConverter.c - wyciąga dane przesłane z serwera w formacie json.
- algorithm.c - zawiera implementację algorytmu umożliwiającego automatyczne przemieszczanie się po mapie.

Wersja do testów posiada analogiczne moduły. Odpowiadają one za testowanie swoich odpowiedników z wersji docelowej programu.

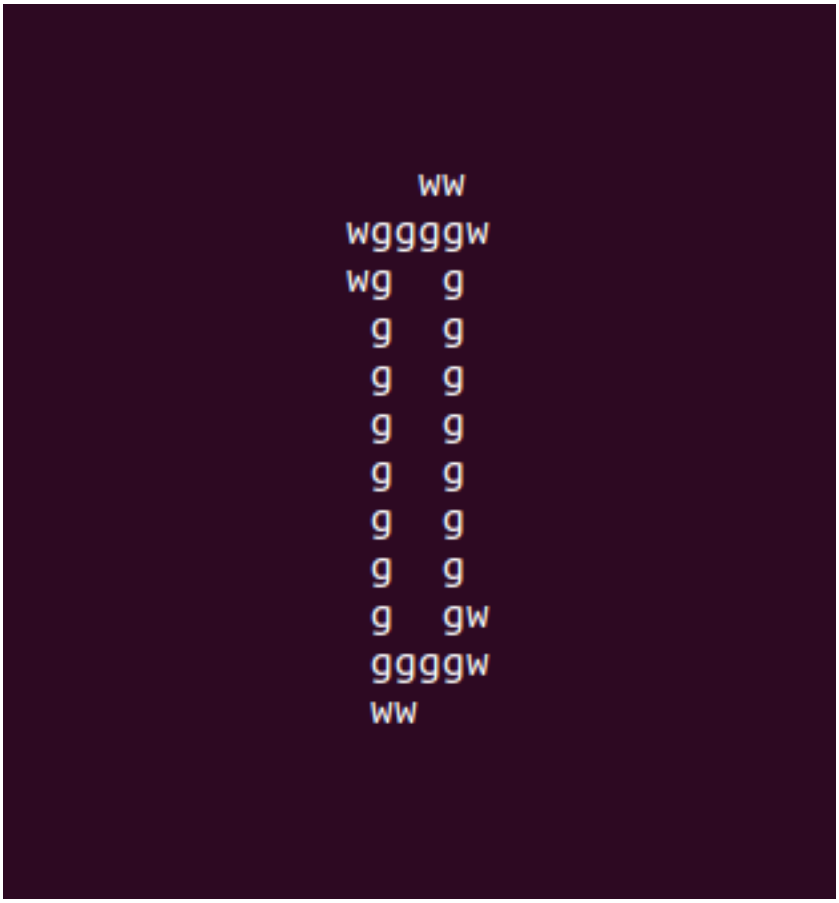
1.4 Mapa

Struktura mapy w programie prezentuje sie następujaco:

```
1
2  struct Map{
3     char*name;
4     intwidth;
5     intlength;
6     intcurrentX;
7     intcurrentY;
8     char currentField; char
9     currentDirection;
      char*fields;
  };
10
```

Przechowuje aktualne informacje o badanym świecie oraz macierz z informacjami o odwiedzonych juz polach.Są to znaki char i posiadają następujące znaczenie:

- g - trawa
- w - ściana

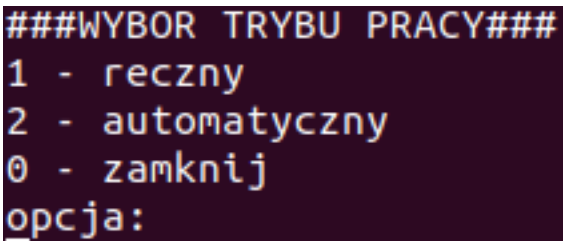


Rysunek 1: Prezentacja mapy w konsoli

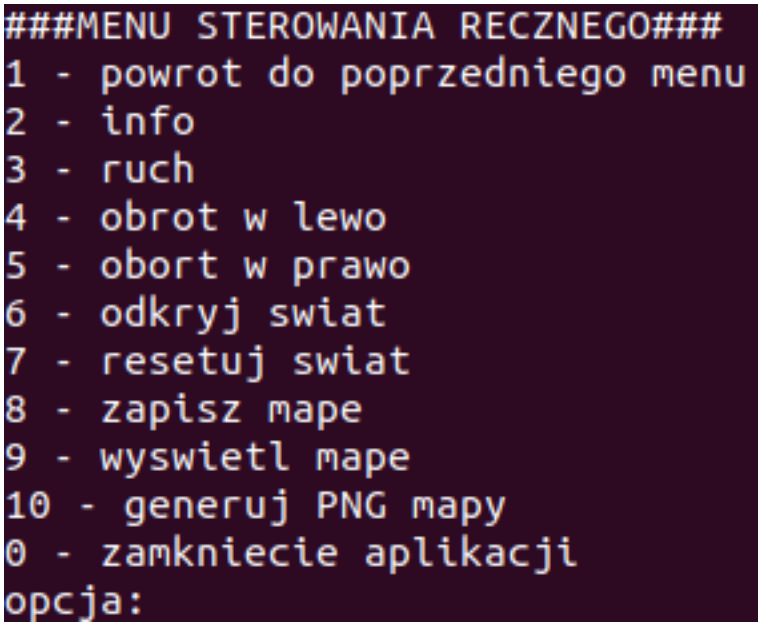
1.5 Menu

Program w celu wygodnego użytkowania został wyposażony w wyświetlane w konsoli menu. Posiada ono dwa poziomy:

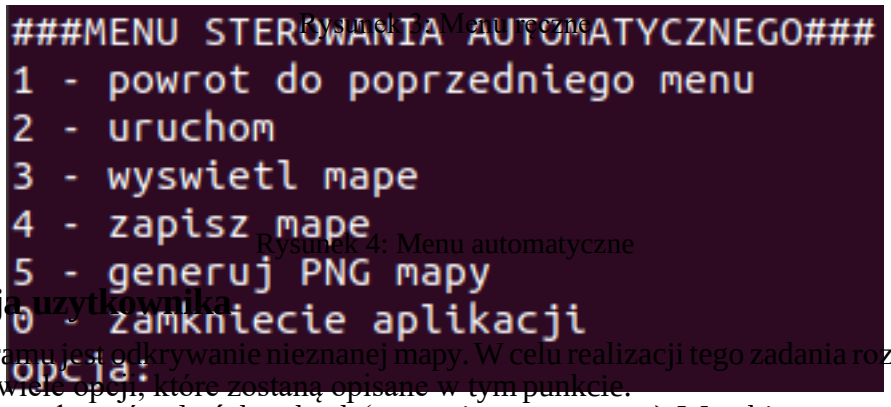
- Na pierwszym poziomie użytkownik wybiera tryb pracy (ręczny lubautomatyczny).
- Po wybraniu trybu pracy wyświetlane są odpowiednie dla niego opcje.



Rysunek 2: Pierwszy poziom menu



Rysunek 3: Menu reczne



Rysunek 4: Menu automatyczne

1.6 Instrukcja uzytkownika

Zadaniem programu jest odkrywanie nieznanej mapy. W celu realizacji tego zadania rozwiązanie zostało wyposażone w wiele opcji, które zostaną opisane w tym punkcie.
Program można użytkować w dwóch trybach (ręczny i automatyczny). W trybie ręcznym można samemu poruszać się po mapie używając odpowiednich klawiszy na klawiaturze:

- 2 - wyświetla informacje o aktualnym położeniu
- 3 - wykonuje ruch do przodu
- 4 - wykonuje obrót w lewo o 90 stopni
- 5 - wykonuje obrót w prawo o 90 stopni

- 6 - wyświetla informacje o sąsiadujących polach
- 7 - resetuje świat
- 8 - zapisuje mapy do pliku .txt
- 9 - wyświetla aktualny stan mapy ze strutkury danych
- 10 - generuje obraz w formacie .png bazując na aktualnym stanie strutkury mapy
- 0 - zamyka aplikacje

W trybie automatycznym użytkownik ma mniejszą ilość opcji do wyboru:

- 2 - uruchamia algorytm odkrywania świata
- 3 - tak jak w trybie ręcznym wyświetla aktualny stan struktury mapy
- 4 - zapisuje aktualny stan strutkruy mapy do pliku .txt
- 5 - generuje obraz w formacie .png
- 0 - zamknięcie aplikacji

1.7 Komunikacja z serwerem

Komunikacja z serwerem została zrealizowana przez implementację odpowiednich funkcji wykonujących zapytania do API. W zależności od trybu pracy programu jego użytkownik lub algorytm wybiera jakie zapytanie chce zadać. Pobierana jest wówczas odpowiedź z serwera w formacie json, która jest już przetwarzana w osobnym module.
Zapytania jakie można wysłać to:

- Info - zwraca informację o obecnym położeniu.
- Move - próbuje wykonać ruch do przodu.
- RotateLeft - wykonuję obrót w lewo o 90 stopni.
- RotateRight - wykonuje obrót w prawo o 90 stopni.
- Reset - resetuje stan świata.
- ExploreWorld - zwraca informację o sąsiadujących polach.

1.8 Odczyt danych z JSONA

Do implementacji parsera do danych w formacie JSON wykorzystana została biblioteka cJSON. W module apiConverter.c została zaimplementowana metoda jsonParserPayload, która przyj- muje w parametrze dane przesłane przez serwer.

```
void jsonParserPayload(char*json_string){
cJSON*name=NULL;
cJSON*root=cJSON_Parse(json_string);

root=cJSON_GetObjectItem(root,"payload");

name=cJSON_GetObjectItem(root,"current_x");
current_x=name->valueint;

name=cJSON_GetObjectItem(root,"current_y");
current_y=name->valueint;

name=cJSON_GetObjectItem(root,"direction");
direction=name->valuestring[0];

name=cJSON_GetObjectItem(root,"field_type");
field_type=name->valuestring[0];
}
```

Poszczególne dane są odczytywane przez odpowiednie metody biblioteki cJSON. Następnie są wykorzystywane w aktualnej strukturze mapy.

1.9 Algorytm

Algorytm zaimplementowany do automatycznego odkrywania świata działa następująco:

1. wykonuje ruch naprzód aż do napotkania ściany.
2. po napotkaniu ściany wykonuje obrót w prawo.
3. wykonujemy jeden ruch do przodu i obrót w lewo.
4. próbujemy wykonać ruch do przodu, jeśli się nie da wykonujemy obrót w lewo i idziemy naprzód.
5. powtarzamy punkty 2-4 aż do powrotu do punktu startowego.

W trakcie wykonywania algorytmu zliczamy ilość obrotów w lewo oraz w prawo. W naszym konkretnym przypadku większa ilość skrętów w lewo oznacza, że znaleźliśmy ścianę wewnętrzną. W tym przypadku rozpoczynamy proces odkrywania świata na nowo. Jeśli jest na odwrót to oznacza, że udało nam się znaleźć otoczkę zewnętrzną badanego świata. W takim przypadku możemy rozpocząć przeszukiwanie pozostalych pól świata.

1.10 Złożoność obliczeniowa

Złożoność obliczeniowa rozpatrywanego problemu w dużej mierze zależy od wymiarów badanego świata. Zakładając najbardziej optymistyczny przypadek, nasz świat ma wymiary n x n i nie posiada żadnych wewnętrznych przeszkód możemy odkryć go wykonując n^2+1 ruchów. Wówczas złożoność obliczeniowa wyniesie $O(n^2)$.
Ten przypadek nie uwzględnia jednak opóźnień jakie mogą być generowane przez oczekiwanie na odpowiedź serwera. W realnym przypadku dla większego rozmiaru n może być to znacząca wartość. Kolejne opoźnienia w realnym przykładzie mogą być generowane przez wewnętrzne przeszkody napotkane podczas eksploracji świata. Koszt odkrycia każdej przeszkody możemy przyjąć jako ilość ruchów konieczną do wykonania w celu jej okrążenia.
W przypadku bardziej skomplikowanego problemu tego typu dobrze mogłyby się sprawdzić algorytmy sztucznej inteligencji.

1.11 Literatura

1. Biblioteka cJSON - https://github.com/DaveGamble/cJSON Data dostępu 18.01.2021
2. Biblioteka curl - https://curl.se/ Data dostępu 19.01.2021
3. Valgrind - https://valgrind.org/ Data dostępu 19.01.2021
4. Libpng - http://www.libpng.org/pub/png/libpng.html Data dostępu 23.01.2021
5. Indent - https://www.gnu.org/software/indent/ Data dostępu 23.01.2021