Michał Odziemkowkski i Kamil Chodorowski

Dokumentacja projektu „Rumpa”

1. Instrukcja obsługi

Na początku gry dobrze jest wykonać funkcję resetującą (make test-reset), sprowadzi to postęp gry do stanu początkowego, synchronizując tym samym stan plików do stanu serwera. Możemy też wyświetlić techniczne informacje zwracane przez serwer (funkcja make test). W konsoli pojawią się informacje o statusie odpowiedzi, nazwie świata, obecnym położeniu na koordynatach x y, kierunek w którym jesteśmy zwróceni, typ pola na którym się znajdujemy, oraz ilość wykonanych przez nas ruchów.
Na tym etapie nic nie przeszkadza nam w rozpoczęciu gry. Mamy dwie opcje:

- Możemy zacząć grę samodzielnie. Tutaj nasza pula możliwości wygląda następująco: możemy obracać się w lewo (make test-rotL), w prawo (make test-rotR), posunąć się o 1 pole do przodu (make test-mov) lub odkryć 3 pola przed nami (pole tuż przed nami i 2 pola na ukos po lewej i po prawej) (make test-exp). Każda z tych komend dodaje 1 ruch do puli wykonanych kroków. Dodatkowo możemy łączyć wiele ruchów za jednym aktywowaniem programu np. „./main qwerty_22 M M E” wykonując w ten sposób 2 ruchy i eksplorację. Dostępne ruchy po „./main”:

  M – ruch do przodu

  E – eksploracja

  Rright – obrót w prawo

  Rleft – obrót w lewo

  reset – resetuje świat

- Zacząć eksplorować mapę automatycznie za pomocą bota wykonując komendę „make test-bot” lub „./main qwerty_22 bot”

2. Podział na pliki:

Program został podzielony na poniższe pliki:

-main.c

Tutaj znajduje się scenariusz działania całego programu. To ten plik decyduje, które ruchy wykona program, ile tych ruchów będzie jak i zleci załadowanie i zapisanie mapy do pamięci stałej w postaci pliku txt. Dzięki takiemu zabiegowi możemy przerwać grę w dowolnym momencie i wrócić do niej później, nawet po restarcie komputera. Program też uruchamia bota.

-mapa.c i mapa.h

Serce całej obsługi naszego świata, w którym toczy się rozgrywka. To tutaj tworzona jest dynamiczna mapa programu(funkcja new_map), która zwiększa się jeżeli gracz dojdzie do krańca wyświetlanego obszaru. Służą do tego funkcje check_border i render. Znajdziemy też funcje obsługujące ruchy czyli: tank_move_whole, tank_rot_whole, tank_exp_whole, tank_res_whole, tank_rot, tank_update, tank_exp i odpowiadające im funkcje wypiujące: write_inf_Map_exp oraz write_info_Map. Do tego dołączona jest wcześniej wspomniana obsługa pliku txt czyli funkcje load i save. Na pewno najważniejszą dla użytkownika funkcją będzie write, ta bowiem wyświetla w przystępnej formie obecny stan mapy. Obsługa pamięci po przejściu całej mapy nie traci ani jednego bajta, co zawdzięczamy funkcjom free_map oraz free_area. Zwalniają one następujące struktury:

Map – działającą jako pamięć podręczna naszego programu. Tu przechowane są informacje o tym jaka jest wielkość mapy gdzie na niej jest jaki typ bloku (trawa, ściana, piasek), obecne położenie czołgu, jego kierunek i delty (delty służą do przesunięcia czołgu w taki sposób aby na początku gry znajdował się na środku, a po rozszerzeniu mapy nadal w tym samym miejscy).

Area i Area3 – struktury pośrednie, do których serwer zwraca informacje w dwóch plikach poniżej opisanych.

-APIdecoder.c i APIdecoder.h

Znajdujemy się w obszarze łączności z serwerem. Tutaj tworzone są adresy url odpowiadające każdemu ruchowi. Mamy zatem funkcje: info, move, rotate, rotatel, explore i reset. Poza tym są jeszcze 2 funkcje obsługi pliku Json czyli DJson_info i DJson_explore. Tutaj tworzone są 2 struktury opisane powyżej (Area i Area3).

-serwer.c i serwer.h

Tutaj trafiają wszystkie adresy stworzone w poprzednim pliku. Wrzucamy je do serwera i dostajemy w zamian informacje potrzebne do wykonania ruchów.

-logika.c i logika.h

Centrum zarządzania botem odkrywającym mapę. Na początku bot szuka ściany, idąc do przodu (funkcja moveToHit). Po uderzeniu w ścianę zmienia taktykę działania. Trzyma się lewej ściany i tworzy obwódkę mapy, i tak aż do położenia początkowego (funkcja roundsL). Przy wykonywaniu tych działań korzysta z funkcji pośrednich getStartX, getStartY, rotL_Move i rotR_Move.

-picture.c i picture.h

Generator pliku png, tworzy obraz mapy i zapisuje go na dysku.

3. Scenariusz działania programu czyli dogłębna analiza funkcji main:

Na samym początku tworzymy potrzebne zmienne char i int, przydadzą się do później wykonywanych czynności. Druga wartość z linii komend odpowiada tokenowi gry, jeżeli łączna ilość poleceń przekazanych do programu jest mniejsza od 3 wypisane zostają informacje serwera. W przeciwnym tworzona jest mapa która aktualizuje się zgodnie z wytycznymi pliku txt. Teraz program sprawdza czy został aktywowany bot. Jeśli nie to
wchodzimy do pętli, wykonywanej tyle razy ile pojawiło się komend ruchu. Ciąg warunków dopasowuje komendę która się zaraz wykona na podstawie wkładu użytkownika. Tutaj może rozegrać się pięć kolejnych scenariuszów odpowiadających kolejno:
-ruchowi
-obrotowi w lewo
-obrotowi w prawo
-eksploracji
-resetowaniu gry
Na końcu pętli program sprawdza położenie czołgu względem granic mapy. Jeśli dojechaliśmy do jednej z nich, mapa zostaje poszerzona dwukrotnie w odpowiednim kierunku.
Zaraz po wyjściu z pętli postęp rozgrywki zapisywany jest do pliku, generuje się plik png z obrazem mapy, a pamięć podręczna w postaci struktury „Map” zostaje zwolniona.
Jeżeli jednak bot został aktywowany, powyżej opisana pętla zostaje pominięta i aktywuje się automatyczna eksploracja algorytmami moveToHit oraz roundsL. Tak jak poprzednio Zapisujemy wynik do pliku txt, png oraz zwalniamy pamięć.

4. Schemat działania programu