Kropki - opis struktury programu

Karol Kuczmarz, 308845

13 lutego 2019

Spis treści

1 Wstęp

Kod źródłowy jest podzielony na 7 modułów, z których korzyszta plik "main.c". W każdym rodziale opiszę poszczególne funkcje i sposób, w jaki działają.

2 Main

1. Zmienne globalne

   (a) _Bool isa
   Rozróżniaa, czy program ma obsługiwać gracza A (wartośc 1) czy gracza B (wartość 0).

   (b) GtkWidget *small_board
   Przechowuje przycisk, którym wybieramy małą planszę.

   (c) GtkWidget *medium_board
   Przechowuje przycisk, którym wybieramy średnią planszę.

   (d) GtkWidget *big_board
   Przechowuje przycisk, którym wybieramy dużą planszę.

2. void i_main_window(GtkWidget *widget)

   Zmienna widget przekazywana do funkcji jest przyciskiem odpowiadającym za wybór wielkości planszy. Funckja uruchamia funkcję "main_window" z odpowiednim parametrem informującym o wielkośći planszy i graczu (A czy B). Ponadto przekazuje do pliku kolejkowego informację o wybranej wielkości planszy.

3. int main(int argc, char *argv[])

   W tej funckji tworzone jest okno startowe, w którym wybiera się wielkość planszy. Wywołuje się także funkcję tworzącą pliki kolejkowe.

3 Moduł board

1. Zmienne globalne

   (a) int map[WIDTH_MAX*HEIGHT_MAX]=0;
   Tablica przechowuje informacje o kropkach na planszy. Opis wartości (gdy podane wartśći zapiszemy z minusem, otrzymamy oznaczenia dla kropek przeciwnika).

      i. 0
      Pusta kropka, którą można zaznaczyć.

      ii. 1
      Zajęta przez gracza, niemająca znaczenia kropka.

      iii. 2
      Kropka, która została otoczona, ale program nie znalazł jeszcze dla niej bazy.

      iv. 3
      Kropka gracza, która jest częścią granicy.

      v. 4
      Kropka przeciwnika, która znajduje się w bazie gracza.

      vi. 5
      Kropka gracza w jego bazie (niekoniecznie narysowanej na ekranie).

      vii. 6
      Kropka gracza w bazie przeciwnika.

      viii. 7
      Niezaznaczona kropka w czyjejś bazie.

   (b) int WIDTH
   Szerokośc planszy powiększona o dwa (wyrażona w liczbie kropek).

(c) int HEIGHT
Wysokość planszy powiększona o dwa (wyrażona w liczbie kropek).

(d) frame my_frame
Zawiera połączenia między sąsiednimi kropkami tworzącymi bazy gracza.

(e) framme opp_frame
Zawiera połączenia między sąsiednimi kropkami tworzącymi bazy przeciwnika.

(f) int opp_points
Przechowuje liczbę punktów przeciwnika.

(g) int my_points
Przechowuje liczbę punktów gracza.

(h) int version
Przechowuje wielkość planszy.

(i) _Bool my_turn
Infomruje, czy teraz jest kolejka gracza.

(j) _Bool small_window_opened
Infomruje, czy jest otworzone małe okno (związane z rozpoczęciem nowej gry bądź zamknięciem gry).

(k) _Bool main_widnow_opened
Informuje, czy gra została już uruchomiona.

(l) GtkWidget *mypoints_info
Przechowuje adres zmiennej GTK_LABEL, która wyświetla liczbę punktów gracza.

(m) GtkWidget *opppoints_info
Przechowuje adres zmiennej GTK_LABEL, która wyświetla liczbę punktów przeciwnika.

(n) GtkWidget *window
Przechowuje oknko rozgrywki.

(o) GtkWidget *whose_turn
Przechowuje adres zmiennej GTK_LABEL, która wyświetla, czy jest teraz tura gracza czy nie.

2. gboolean button_press_event(GtkWidget *widget, GdkEventButton *event, gpointer data)

   Gdy jest teraz kolejka gracza oraz nie ma otwartego żadnego okna pomocniczego (np. o rozpoczęciu nowej gry), sprawdza, czy kliknięcie trafia w kropkę. Jeśli trafi i ruch jest legalny, to wyoknuje, wywołuje rysownie planszy oraz gdy już nie ma więcej pustych kropek, kończy grę.

3. gboolean create_map(GtkWidget *widget, cairo_t *cr, gpointer user_data)

   Rysuje planszę. Najpierw linie tworzące ewentualne bazy, a potem kropki w odpowiednich kolorach.

4. void new_game(_Bool isA)

   Rozpoczyna nową grę. Zeruje zmienne globalne takie jak punkty oraz usuwa informacje o połączeniach między kropkami tworzącymi bazy.

5. int opponents_move(gpointer data)

   Zarządza informacjami uzyskiwanymi z pliku kolejkowego. Gdy przeciwnik wykonał ruch, obsługuje go wywołując funkcje analizy ruchu i rysowania. Przetwarza też informacje o wielkoścli planaszy przeciwnika i o ewentualnym opuszczeniu przez niego gry.

6. void points_update(void)

   Wpisuje do zmiennym GTK_LABEL mypoints_info i opppoints_info zaktualizowane liczby punktów.

7. void new_game_window(GtkWidget *widget, gpointer data)

   Otwiera okno, w którym użytkownik podejmuje decyzję o rozpoczęciu nowej gry, czyli poddaniu rozgrywki i powrtou do okna startowego.

8. gboolean close_main_window(GtkWidget *widget, gpointer data)

   Zamyka okno z grą, przy okazji tekża okno, w którym podejmowało się decyzję o opuszeczeniu gry. Ustawia zmienne globalne informujące o tym, czy duże/małe okno jest otwarte tak, by informawały, że nie jest.

9. gboolean close_small_window(GtkWidget *widget, gpointer data)

   Zamyka małe okno (związane np. z ropoczęciem nowej gry).

10. void main_window(int size)

    To jest funkcja, która tworzy okno z grą. Najpierw ustawia zmienne WIDTH, HEIGHT i version na odpowiednie względem argumentu wywołania programu. Później tworzy odpowiednie zmienne GTK_WIDGET i pakuje je do GTK_GRID. Przypisuje funkcje odpowiednim zdarzeniom.

11. void close_main_window_alert()

    Otwiera okno, gdy gracz chce opuścić grę, prosząc go o potwierdzenie swojej decyzji. Jeżeli gracz potwierdzi, to zamyka cały porgram.

12. void check_version(int sign)

    Sprawdza, czy podany jako argument rozmiar planszy przeciwnika zgadza się z rozmiarem planszy gracza. Jeżeli nie, to zamyka program i informuje o błędzie.

13. _Bool the_end(void)

    Sprawdza, czy na planszy są jeszcze jakieś puste kropki, które można zaznaczyć.

14. void end_game_window(void)

    Fukcja otwiera okno, w którym informuje o tym, kto wygrał i podaje zdobyte przez graczy punkty. Następnie zamyka okno z grą.

15. void update_whose_turn()

    Zmienia odpowiednio zmienną whose_turn tak, by informwała, czyja jest teraz kolej.

16. void inform_opp(GtkWidget *widget, gpointer data)

    Wysyła informację do pliku kolejkowego o tym, że gracz opuścił grę.

17. void opponent_left(void)

    Otwiera okno, w którym informuje, że przeciwnik się poddał. Pokazuje wynik a później zamyka okno z grą.

4 Moduł move

1. void move(int map[], frame *lines, int chosen, int *opp_points, int *my_points, int const WIDTH, int const HEIGHT)

   Wywołuje po kolei odpowiednie funkcje z modułu base, przekazując odpowiednio uproszczone wersje planszy.

2. void pointsandverify(int map[], int mapcopy[], int mapcopy2[], int *opp_points, int *my_points, int const WIDTH, int const HEIGHT)

   Porównuje uproszczoną planszę powstałą w fukcji move z oryginalną. Przyznaje punkty.

5 Moduł base

1. Struktury

   (a) Polygon
   Zawiera kolejne wierzchołki łamanej w układzie współrzędnych. Zmienna num informuje o liczbie wierzchołków w łamanej a tablica node zawiera kolejne wierzchołki.

   (b) Segment
   Zawiera całkowite współrzędne końców odcinka.

(c) Frame

Zaweria informację o tym, które wierzchółki tworzą ze sobą bazę - dokładniej tablicę struktur Segment zawierającą wierzchołki, które trzeba ze sobą połączyć. Zmienna num mówi, ile jest takich odcinków.

2. void base(int maporg[], int map[], int map_help[], poly res, int beg, int act, int const WIDTH, int const HEIGHT)

   Poprzez przeszukiwanie z nawrotami tworzy wszystkie możliwe łamane z punktu, który został zaznaczony. By zmniejszyć złożoność korzysta z algorytmu miotły (gdy dwa odcinki w łamanej się przecinają w punkcie o niecałkowitych współrzędnych nie brnie w tę gałąź rekurencji). Gdy trafi z powrotem do punktu wyjścia wywołuje funckję, która sprawdza, które punkty są wewnątrz wielokąta.

3. void push_poly(poly a, int number)

   Dodaje do struktury Polygon nowy wierzchołek. Wstawia go do pierwszego wolnego indekksu w tablicy.

4. int pop_poly(poly a)

   Usuwa ostatni element tablicy w strukturze Polygon.

5. _Bool sweep(poly test, int const WIDTH, int const HEIGHT)

   Algorytm miotły. Najpierw tworzy tablicę odcinków wchodzącą w skład krzywej. Później przez układ współrzędnych przepuszcza poziomą linię. Gdy odcinek leży na tej linii, to jest dodawany do puli odcinków, które sprawdza się, czy wzajmenie się ze sobą krzyżują.

6. _Bool ifcross(seg a, seg b)

   Sprawdza, czy dwa odcinki krzyżują się ze sobą w punkcie nie będącym końcem któregoś z odcinków. Korzysta w wyznacznika.

7. _Bool ifinside(seg polynoid[], poly test, int point, int const WIDTH, int const HEIGHT)

   Sprawdza, czy dany wierzchołek leży wewnątrz wielokąta. Prowadzi poziomą kreskę przez punkt i liczy liczbę przecięć z wielokątem po prawej i lewej stronie.

8. void checkinside(poly res, int maporg[], int map[], int map2[], int const WIDTH, int const HEIGHT)

   Tworzy tablicę odcinków wchodzących w skład wielokąta. Dla każdego punktu z planszy poprzez funkcję ifinside sprawdza, czy punkt leży wewnątrz. Jeżeli tak, to zmienia jego wartość w tablicy.

9. void buildbase(frame *lines, int map[], int const WIDTH, int const HEIGHT)

   Dla każdego punktu z planszy potrzebującego bazy uruchamia funkcję szukającą tej bazy.

10. void findbase(frame *lines, int map[], int index, int const WIDTH, int const HEIGHT)

    Szuka bazy dla punktu. Szuka wielokąta o jak najmniejszym polu, który będzie tworzył bazę dla tego punktu. Jeżeli punkt poziomo lub pionowo sąsiaduje z punktem także potrzebującym otoczenia, to fukcja dla takiego punktu też jest uruchamiana.

11. void push_frame(frame *lines, int n1, int n2, int map[], int const WIDTH, int const HEIGHT)

    Dodaje do struktury Frame nowe odcinki wchodzące w skład bazy. Jeżeli te odcinki już są, to nie dodaje ich drugi raz. Pierwsze współrzędne odcinka na planszy są mniejsze niż drugie.

12. int det(int x1, int y1, int x2, int y2, int x3, int y3)

    Liczy wyznacznik dwóch wektórów na płaszczyźnie.

13. int sgn(int x)

    Funckja signum.

14. void swap(int *a, int *b)

    Zamienia wartości dwóch zmiennych typu int.

6 Moduł communication

1. Zmienne globalne

   (a) PipesPtr Channel

   Zawiera informacje o plikach kolejkowych i o wersji programu (A czy B).

2. void fifo_init(int argc, char *argv[])

   Przypisuje zmiennej globalnej Channel adres struktury pipes z informacjami o plikach kolejkowych i wersji programu (A czy B).

3. void send_info(int index)

   Otrzymuje zmienną int i wypisuje ją do tablicy charów text. Następnie adres tablicy jest przekazywany do funkcji wypisującej tablicę do pliku kolejkowego.

4. void get_info(int *ans)

   Sprawdza, czy program odczytał jakieś informacje z pliku kolejkowego. Gdy pierwsza cyfra jest liczbą, to wczytuje całą liczbę, gdy pierwsza cyfra nie jest liczbą, ale wczytany teskt jest dłuższy niż 1, to odczytuje liczbę od drugiego indeksu tablicy. Gdy nie otrzymano żadnej informacji ustawia wartośc przekazywanej zmiennej na -1.

5. int string_to_int(char text[])

   Zamienia liczbę wpisaną do tablicy charów na zmienną typu int.

7 Moduł fifo

1. Struktury

   (a) pipes

   Zawiera wskaźniki na struktury FILE dotyczące pliku, z którego czytamy i pliku, do którego wysyłamy informacje. Ponadto zawiera zmienną typu _Bool, która mówi, czy program obsługuje gracza A.

2. void closePipes(PipesPtr pipes)

   Zamyka plik, z którego czytamy i plik, do którego wysyłamy informacje.

3. PipesPtr initPipes(int argc,char *argv[])

   Zwraca adres na strukturę pipes, czyli tworzy pliki kolejkowe i zaznacza wersję programu (A czy B). W przypadku niepowodzenia wyisuje błąd i zamyka program.

4. static FILE *openOutPipe(char *name)

   Otwiera plik kolejkowy, z którego czytamy informacje

5. static FILE *openInPipe(char *name)

   Otwiera plik kolejkowy, do którego wysyłamy informacje.

6. void sendStringToPipe(PipesPtr pipes, const char *data)

   Wypisuje odpowiednią informację do pliku kolejkowego.

7. _Bool getStringFromPipe(PipesPtr pipes, char *buffer, size_t size)

   Wczytuję jedną linię z pliku koljekowego do tablicy charów. Gdy nic nie wczytano zwraca 0, w przeciwnym wypadku 1.

8 Moduł coordinates

1. Struktury

   (a) Vector

   Zawiera całkowite współrzędne na płaszczyźnie. Najpierw współrzędno osi OX a potem OY.

2. vec dot_clicked(double x, double y)

   Sprawdza, czy kliknięcie trafiło w jakąkolwiek kropkę. Zwraca współrzędne tej kropki lub punkt (-1, -1), gdy nic nie trafiono.

3. _Bool verify_click(int map[], vec test, int const WIDTH, int const HEIGHT)

   Sprawdza, czy przeciwnik trafił w odpowiednią kropkę, tzn. czy może ją zaznaczyć.

9 Moduł definitoins

Ten moduł składa się tylko z pliku nagłówkowego, w którym są załączone wszystkie potrzebne biblioteki oraz polecenia "DEFINE", które ustalają promień kropek (RADIOUS), przerwę między nimi (GAP) oraz maksymalną wielkość planszy (MAX_WIDTH - szerokość, MAX_HEIGHT - wysokosc).