Dokumentacja techniczna

Projekt G6 - „kółko i krzyżyk”

Autor: Marcin Trajfacki

301199

Opis projektu

Projekt G6 to zmodyfikowana wersja gry kółko i krzyżyk. Jest to turowa gra dwuosobowa. Napisana została w języku C, w standardzie C99, przy użyciu interfejsu okienkowego GTK+3.0. Korzysta również z gotowego programu „gtk-talk”. Jego działanie jest szerzej opisane w sekcji „Komunikacja”. Do działania projektu wymagana jest biblioteka „gtk+-3.0”.

Spis treści

Podział na moduły

Program podzielony jest na 4 pliki źródłowe oraz 1 plik nagłówkowy:

- main.c
- board_drawing.c
- game_logic.c
- communication.c
- tic_tac_toe.h

main.c

Plik main.c odpowiada za utworzenie okna w interfejsie GTK, dostępnego z poziomu całego projektu- każdy plik, po zaimportowaniu modułu „tic_tac_toe.h” może modyfikować zawartość okna. Ponadto odpowiada on za obsługę błędów wywołanych nieprawidłowymi danymi wejściowymi podawanymi przy uruchomieniu programu. Zdefiniowane tam zostały również dwie funkcje: kill_process() oraz pokazBlad().

- **void kill_process()**

  Funkcja ta odpowiada za zamknięcie dostępu do plików kolejkowych określonych jako pipes→fifo_in oraz pipes→fifo_out. Ponadto zwalnia ona pamięć do nich przypisaną. Zamyka również główne okno programu, wywołując funkcję gtk_main_quit().

- **void pokazBlad(char *komunikat)**

  Jest to funkcja należąca do modułu „gtk-talk”. Jej deklaracja została jednak przeniesiona do pliku main.c. Odpowiada ona za utworzenie nowego widgetu, będącego oknem wyświetlającym podany przez użytkownika jako parametr tekstu.

board_drawing.c

Plik board_drawing.c zawiera definicje dwóch funkcji: create_grid() oraz board_update(). Ponadto inicjalizowane są w nim zmienne odpowiedzialne za przechowywanie danych gry oraz tablicę dwuwymiarową pól gry.

- void create_grid()

Powyższa funkcja modyfikuje utworzone w programie main.c okno gry, dodając do niego dwuwymiarową planszę o rozmiarze określonym przez użytkownika jako argument wywołania programu. Każde pole utworzonej planszy jest typu GtkWidget, a następnie wywołana na nim jest funkcja gtk_button_new_with_label, przez co każde z pól staję się przyciskiem. Każdy z przycisków, kliknięty, wywołuje funkcję click_parser(), przekazując jej dane o samym sobie. Wykorzystuje do tego strukturę danych field.

- struktura field

Ta struktura danych wykorzystywana jest do przekazywania informacji o kliknięty przycisku. Przechowuje dwie zmienne typu int – x oraz y, oraz zmienną char sign. Odpowiedzialne są one za określenie pozycji kliknięty przycisku w siatce przycisków oraz znak do niego przypisany.

- void board_update()

Wywoływana cyklicznie jako fragment funkcji update_data(), odpowiedzialna jest ona za sprawdzenie stanu danych gry w tablicy game_data, oraz naniesienie ich na siatkę przycisków poprzez przypisanie im odpowiadającego im w danych gry symbolu.

game_logic.c

Funkcje zdefiniowane w tym pliku odpowiadają za przetwarzanie kliknięć, sprawdzanie czy któryś z graczy wygrywa oraz przygotowanie danych do wysłania.

- void click_parser(GtkWidget *widget, gpointer user_data)

Powyższa funkcja wywoływana jest, jeśli kliknięty zostanie jakikolwiek przycisk na siatce przycisków. Na podstawie jego pozycji oraz znaku określa, jaki ruch powinien zostać wykonany. Ponadto, przed wykonaniem jakiegokolwiek ruchu

wywołuje funkcję segfault_protector(). Jeśli po żądanej operacji wystąpiłoby naruszenie pamięci, wywołana zostaje funkcja pokazBlad(), a click_parser wraca do miejsca wywołania. Jeśli jednak jakiś ruch zostanie wykonany, wywołuje ona funkcję send_data().

- void set_on_bottom(int column, coordinates *data)

Po wywołaniu, na dole kolumny określonej zmienną column postawiony zostanie przypisany do gracza znak.

- void set_on_top(int column, coordinates *data)

Funkcja ta postawi na pierwszym wolnym od góry polu w kolumnie column znak odpowiadający klikającemu graczowi.

- void complex_move(int column, coordinates *data)
Ta funkcja wykonuje ruch opisany jako komenda ‘b’ w dokumentacji użytkownika. Wyciąga znak z dołu kolumny, przesuwa ją w dół, kładzie wyciągnięty znak na wierzchu kolumny oraz dokłada na szczycie w ten sposób powstałej kolumny znak gracza.

- void send_data()

Odpowiada za przygotowanie danych gry do zapisu w pliku kolejkowym. W tym celu zapisuje char sign przechowywane w tablicy game_data w tablicy typu char. Następnie dodaje na koncu znak ‘\0’, po czym wywołuje funkcję sendStringToPipe() z powstałą w ten sposób tablicą znaków jako argument.

- gboolean update_data(gpointer data)

Powyższa funkcja odpowiada za przeczytanie danych zapisanych w pliku kolejkowym, które zapisane tam zostały przez drugiego gracza. Interpretuje te dane i zapisuje je w tablicy game_data, po czym wywołuje funkcję board_update(). Wywoływana jest cyklicznie, więc zawsze zwraca prawdę.

- void win_loop(gpointer data)

Funkcja wywoływana jest cyklicznie w funkcji main(). Jej przeznaczeniem jest wywołanie funkcji check_for_win() dla obydwu symboli graczy (‘X’ oraz ‘O’), oraz, w razie jakiegokolwiek zwycięstwa, wyświetlenie odpowiedniego komunikatu oraz wywołanie funkcji kill_process().

- bool check_for_win(char player)

Celem tej funkcji jest iteracja po całej tablicy dwuwymiarowej game_data oraz zliczenie występujących po sobie takich samych symboli podawanych jako jej

argument. Jeśli w pionie, poziomie, lub w którejś z przekątnych wystąpi określona przez gracza jako rozmiar planszy ilość takich samych znaków, zwraca prawdę; w przeciwnym przypadku zwraca fałsz.

- static bool segfault_protector(int column)

Działanie tej funkcji polega na zliczeniu wolnych miejsc w podanej jako parametr kolumnie w danych gry. Jeśli niemożliwe jest dołożenie kolejnego symbolu gracza, zwraca prawdę. W przeciwnym przypadku zwraca fałsz.

communication.c

Plik communication.c zawiera funkcje zdefiniowane w programie „gtk-talk". Odpowiada on plikowi lin-fifo.c. Zawiera funkcje void closePipes(PipesPtr pipes), PipesPtr initPipes(int argc, char *argv[]), static FILE *openOutPipe(char *name), static FILE *openInPipe(char *name), void sendStringToPipe(PipesPtr pipes, const char *data) oraz bool getStringFromPipe(PipesPtr pipes, char *buffer, size_t size). Ponadto zdefiniowana jest w nim struktura danych pochodząca z „gtk-talk" - struct pipes.

tic_tac_toe.h

Plik nagłówkowy tic_tac_toe.h zawiera deklaracje prawie wszystkich wymienionych wyżej funkcji, nie licząc funkcji segfault_protector().

Ponadto zadeklarowane są w nim zmienne współdzielone pomiędzy plikami programu, takie jak extern GtkWidget grid, extern GtkWidget *buttons[11][11], extern int size, extern field *game_data[11][11], extern char player indicator, extern char this_player_turn.

Zawarte są w nim również definicje struktur coordinates oraz field. Struktura field przechowuje zmienne int x, int y oraz char sign, zaś coordinates zawiera dodatkowo GtkWidget *clicked_button.

Na końcu pliku, po komentarzu //communication znajduje się również zawartość pliku fifo.h z programu „gtk-talk".

Komunikacja

Potrzeba wykorzystania komunikacji między procesami wynika z faktu, że projekt G6 to gra dwuosobowa. Każdy gracz uruchamia własną kopię gry, więc celem współdzielenia danych wymagana jest komunikacja między tymi kopiami. Do tego celu użyte zostały pliki kolejkowe. Tworzone przy pierwszym uruchomieniu programu pliki AtoB oraz BtoA zawierają dane wymieniane przez obie kopie gry. Ponadto, określona przez gracza A jako parametr wywołania wartość size przekazywana jest graczowi B również za pomocą pliku kolejkowego. Komunikacja możliwa jest dzięki wykorzystaniu fragmentów programu „gtk-talk”, będącego komunikatorem tekstowym. Po otwarciu kanału komunikacji między procesami ich wymiana następuje sekwencyjnie, to znaczy jeśli jeden z graczy wykonał już jakiś ruch, przesyłając tym samym dane, musi zaczekać na otrzymanie danych od drugiego gracza, zapisana jest tam bowiem również informacja pozwalająca graczowi na wykonanie kolejnego ruchu. Przy odczytywaniu danych z pliku zapisywana jest ona w zmiennej this_player_turn, zaś po wysłaniu jej wartość się zmienia, uniemożliwiając wykonanie kolejnego ruchu, przez co jest swego rodzaju żetonem.