Projekt - gra Memory

Wstęp do programowania w języku C

Michał Górniak

1 lutego 2018

1 Zasady gry

Gra Memory jest grą przeznaczoną dla dwóch osób (choć możliwa będzie też gra jednoosobowa). Gracze wykonują ruchy na przemian. W każdym ruchu aktulany gracz wybiera dwie spośród wszystkich zakrytych na planszy kart i odkrywa je. Jeśli obie karty mają ten sam obrazek, gracz pozostawia te karty odkryte i zdobywa punkt, w przeciwnym przypadku zakrywa karty. Gra kończy się, gdy wszystkie karty zostaną odkryte, a wygrywa gracz, który zdobył więcej punktów.

2 Opis programu

2.1 Menu główne

Po uruchomieniu programu użytkownik zobaczy menu główne. Będzie ono miało formę listy dostępnych opcji. Użytkownik porusza się po menu strzałkami, a aktualnie wybrana opcja jest delikatnie podświetlona. Wśród nich znajdą się: **Nowa gra**, **Kontynuuj rozpoczętą grę**, **Rekordy**. W prawym górnym rogu będzie umieszczony przycisk **X**, którego wciśnięcie zakończy program.

Po wybraniu nowej gry, gracz zostanie poproszony o wybór trybu gry: **Gra na czas** (gracz nie ma przeciwnika), **Rozgrywka z komputerem** (przeciwko graczowi gra komputer) oraz **Rozgrywka z przeciwnikiem**.

Przy rozpoczynaniu nowej gry gracz może również wybrać jeden z kilku dostępnych rozmiarów plansz oraz tematykę obrazków.

Jeśli gracz chce kontynuować rozpoczętą rozgrywkę, to wybiera stan gry z listy dostępnych zapisów – maksymalnie 10 zapisanych stanów gry.

W rekordach zapisane będą najepsze wyniki gracza z trybu gry na czas.

2.2 Rozgrywka

2.2.1 Gra na czas

W trybie treningowym gracz cały czas odkrywa po dwie karty i jeśli okażą się one takie same, to zostają odkryte. Na górze ekranu wyświetla się czas od rozpoczęcia rozgrywki. Gra kończy się, gdy gracz odkryje wszystkie karty.

2.2.2 Rozgrywka z przeciwnikiem

Gracze na przemian wykonują ruchy zgodne z zasadami gry. Wygrywa gracz, który na koniec gry zdobył więcej punktów.

2.2.3 Rozgrywka z komputerem

Rozgrywka wygląda identycznie jak w przypadku gry z przeciwnikiem, tylko ruchy za drugiego gracza wykonuje komputer.

3 Implementacja

Projekt został podzielony na trzy części.

3.1 main

Krótki program z jedną funkcją main, która uruchamia właściwe okienko z modułu GTK.

3.2 gtk-lib

Główny moduł projektu. Odpowiada za wszystkie graficzne części Memory oraz za samą rozgrywkę.

Jest podzielony na następujące funckje:

1. void quit(GtkWidget *widget, gpointer data) - wyłącza program
2. GtkWidget *button_constructor(char *label, void (*function)(GtkWidget*, gpointer), gpointer data) - zwraca wskaźnik na gotowy przycisk z zadanym napisem i funkcją oz danymi, która wykonuje się po wciśnieciu tego przycisku
3. void button_swap(GtkWidget *button, int height, int type, int number) - zmienia stan pola w grze z odkrytego na nieodkryte i z powrtoem
4. GtkWidget *label_constructor(char *text) - tworzy label z określonym napisem
5. GtkWidget *text_constructor(int max_lenght) - tworzy pole do wpisywania tekstu z zadaną maksymalną długością wpisywanego tekstu
6. info *info_constructor(game_data *game, GtkWidget *widget, int type) - konstruktor do struktury info
7. move_info *move_info_constructor(game_data *game, GtkWidget *window, GtkWidget ***buttons, GtkWidget *label, GtkWidget *label2, char *text, int i, int j) - konstruktor do struktury move_info
8. char *get_text(GtkWidget *text, int max_lenght) - zwraca napis z widgetu typu GTK_ENTRY
9. GtkWidget *get_window() - tworzy okno z domyślnymi parametrami
10. void configurate_window(GtkWidget *window, GtkWidget *box, GtkWidget *grid) - konfiguruje ustawienia okna
11. void set_size(GtkWidget *widget, gpointer data) - ustawia rozmiar planszy przy wyborze rodzaju gry
12. void set_images(GtkWidget *widget, gpointer data) - ustawia rodzaj obrazków przy wyborze rodzaju gry

13. void random_shuffle(int *array, int n) - losowo permutuje tablicę array o n elementach
14. void prepare_tiles(game_data *game) - przygotowuje losową planszę z zadanami parametrami gry
15. char *int_to_char(int x) - zamienia inta na char*
16. char *seconds(int x) - zwraca odpowiednią odmianę słowa sekunda w zależności od x
17. int update_time(gpointer data) - uaktualnia czas w trybie "gry na czas"
18. void run_timer(GtkWidget *widget, gpointer data) - uruchamia mierzenie czasu w trybie "gry na czas"
19. void stop_timer(GtkWidget *widget, gpointer data) - zatrzymuje mierzenie czasu w trybie "gry na czas"
20. int make_second_move_computer(gpointer data) - wykonuje drugi ruch komputera w "grze jednoosobowej z komputerem"
21. int make_move_computer(gpointer data) - wykonuje pierwszy ruch komputera w "grze jednoosobowej z komputerem"
22. GtkWidget *save_frame(int number, gpointer data) - tworzy ramkę ze slotem do zapisu gry
23. int run_wait(gpointer data) - funkcja potrzebna do zatrzymania gry w trybie "gry na czas"
24. void make_move(GtkWidget *widget, gpointer data) - wykonuje ruch po wciśnięciu odpowiedniego pola
25. void run_game(GtkWidget *widget, gpointer data) - uruchamia grę z zadanymi parametrami
26. void run_make_save(GtkWidget *widget, gpointer data) - uruchamia okienko do zapisu stanu gry
27. void run_select_game(GtkWidget *widget, gpointer data) - uruchamia okienko do wyboru trybu gry
28. void run_records(GtkWidget *widget, gpointer data) - uruchamia rekordy z trybu "gry na czas"
29. void run_menu(GtkWidget *widget, gpointer data) - uruchamia menu główne
30. void run_game_over(GtkWidget *widget, gpointer data) - uruchamia okienko wyświetlające się po skończeniu gry
31. void run_new_game(GtkWidget *widget, gpointer data) - uruchamia okienko wyboru "Nowej gry"

3.3 saves-lib

Moduł odpowiadający za zarządzanie plikami gry. Obsługuje zapisy stanu gry oraz zapisy rekordów.

Jest podzielony na następujące funkcje:

1. void convert_to_file(game_data *g, char *filename) - konwertuje stan gry do pliku o zadanej nazwie
2. game_data *convert_to_game(char *filename) - konwertuje stan gry z pliku o zadanej nazwie
3. char *info_about_file(char *filename) - zwraca informacje o zapisie – trybie gry, graczach i dokładnej dacie zapisu
4. record_info *record_constructor(int seconds, char *name, char *date1, char *date2) - konstruktor struktury record_info
5. bool record_compare(record_info *a, record_info *b) - porównuje dwa wyniki i wybiera lepszy
6. void update_records(game_data *game) - uaktualnia rekordy po skończeniu każdej gry w trybie "gry na czas"
7. GtkWidget *get_records(int height) - zwraca label z pełnym rankingiem rekordów dla planszy o zadanych wymiarach