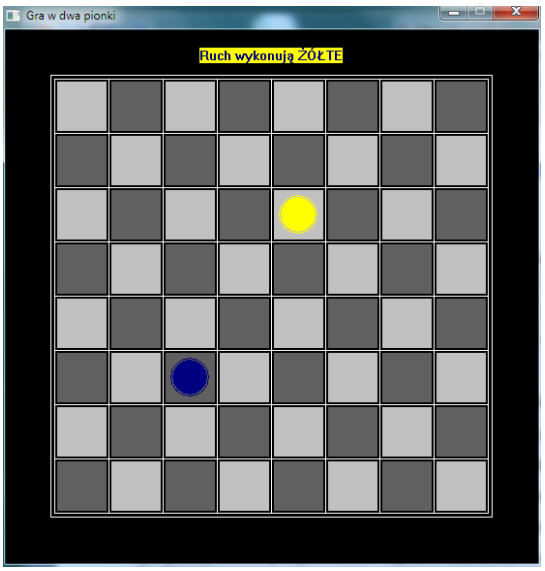
Środowisko Dev-C++ i biblioteka WinBGIm.

Zadanie



Napisać prostą grę, w której na szachownicy zawierającej 64 pola o kolorach jasnoszarym i ciemnoszarym poruszają się dwa pionki: niebieski i żółty. Każdy pionek, w przypadającej na niego kolejce ruchu może zostać przesunięty na jedno z ośmiu sąsiadujących z nim pól. Pionki nie mogą wychodzić poza szachownicę. Gracze naprzemiennie wykonują ruchy wskazując kliknięciem myszki, gdzie pionek ma się przesunąć. W przypadku kliknięcia poza obszarem szachownicy lub w miejsce, na które pionek nie może być przesunięty, gracz zostaje o tym poinformowany odpowiednim napisem. Gra nie kontroluje czy pionki stawiane są jeden na drugim czy nie.

W dowolnym momencie, po wciśnięciu klawisza [S] na klawiaturze, gra pozwala zapisać obraz stanu gry w postaci pliku graficznego na dysku.

Szachownica rysowana jest w oknie o rozmiarach 550 x 550 pikseli i ograniczona kwadratem o wierzchołkach w pozycjach: (46,46) i (502,502). Pola w kolorach jasno i ciemno szarym oddalona są od białych krawędzi o 2 piksele z każdej strony.

Cel

Poznanie środowiska Dev-C++ (wxDev-C++ bez obsługi wxWidgets) oraz zapoznanie się z możliwościami biblioteki WinBGIm w tym dostępnymi funkcjami graficznymi oraz obsługą myszy i klawiatury.

Środki

Środowisko wxDev-C++ , biblioteka WinBGIm.

Zarys możliwego rozwiązania

Na początek trzeba sobie wyznaczyć ogólne wzory na położenie wierzchołków poszczególnych pól szachownicy. Wzory takie ułatwią napisanie poprawnych pętli. Na pewno wielokrotnie będziemy musieli poszukiwać środka pola szachownicy znajdującego się najbliżej jakiegoś punktu (x,y). Na przykład w tedy, gdy klikamy myszą i chcemy wiedzieć w jakim punkcie ma się znaleźć środek rysowanego okręgu będącego pionkiem. Dlatego, dobrze by było stworzyć sobie odpowiednią do tego celu funkcję. W jej wnętrzu też skorzystamy z opracowanych wcześniej wzorów. Dlatego naprawdę warto usiąść z ołówkiem i kartką papieru i je wyprowadzić.

Główną część programu można zrealizować na wiele sposobów. Można stworzyć pętlę oczekującą na wciśnięcie klawisza myszy i dopiero wówczas sprawdzać czy dany ruch jest możliwy do wykonania. Można również stworzyć pętlę, która zawsze czeka na wykonanie jakiegoś ruchu, a samą obsługę wykonania ruchu przenieść do handlera kliknięcia myszą. Wówczas wszystkie procedury sprawdzające wywoływane będą automatycznie przez system, w chwili kliknięcia. Można także stworzyć dwa obiekty będące pionkami, które będą odbierały sygnały od myszy i odpowiednio na nie reagowały. Zapewne istnieje jeszcze wiele innych rozwiązań[1].

Drobnego wyjaśnienia może wymagać zanikanie komunikatów wyświetlanych na górze okna w kilka sekund po ich pojawieniu się. Można to zrobić poprzez stworzenie licznika, który inicjalizowany jest pewną wartością w chwili wyświetlenia komunikatu, którą następnie zmniejsza się w głównej pętli programu lub procedury obsługującej oczekiwanie na ruch. Po osiągnięciu wartości zero przez licznik, komunikat przestaje być wyświetlany.

Przygotowany fragment programu

Przygotowane źródła zawierają jedynie sekcję dołączania plików nagłówkowych, częściowo wypełnioną funkcję **`main()`** oraz pustą funkcję **`void save_file()`** i niemal pustą funkcję **`void draw_dashboard()`** rysującą ramkę wokół szachownicy.

Jak się przygotować przed zajęciami

Przed zajęciami proszę zapoznać się z następującymi zagadnieniami: inicjalizacja okna w WinBGIm, wykonywanie podstawowych operacji graficznych w tej bibliotece (rysowanie punktu, odczytywanie koloru punktu, rysowanie linii, kopiowanie wycinka obrazu etc). Należy zapoznać się również z metodami obsługi myszy w bibliotece, ze szczególnym uwzględnieniem wykorzystania funkcji *handlerów*.

[1] Ja w przykładowym programie zastosowałem to drugie rozwiązanie (J.T.).