ALQUERQUEgame

By Konrad Zegar

INSTRUKCJA

Grę uruchamiamy od okna menu gdzie mamy trzy prowadzące przyciski:

- Play – uruchamia okno w którym wybieramy tryb gry (dwuosobowy i jednoosobowy) wpisujemy również swoje nazwy, później możemy startować przyciskiem start

- Ranking – uruchamia listę rankingową wcześniejszych graczy, lista pokazuje 10 najlepszych wyników sortowanych po liczbie ruchów i czasie rozgrywki

- Exit – przez naciśnięcie lewego klawisza myszy na przycisku pojawia się małe okno potwierdzenia wyjścia do pulpitu gdzie mamy dwie opcje wyjść albo nie.

Od początku rozgrywki uruchamiany jest timer. Rozpoczyna gracz z zielonymi pionkami.

Sposób poruszania

W swojej turze gracz myszką nakierowuje na wybranego swojego pionka i klika LPM (lewy przycisk myszy) po wybraniu pojawiają się na żółto możliwe ruchy, wybrany ruch również klikamy LPM i tak pionek przemieszcza się na nowe miejsce. W przypadku gdy chcemy anulować wybór pionka klikamy PPM (prawy przycisk myszy) i wybieramy innego pionka.

Reguły gry

W grze przemieszczamy się pionkami po planszy według linii po jednym ruchu, pionki przeciwnika możemy zbijać poprzez przeskoczenie. Gra kończy się gdy któryś z graczy straci wszystkie swoje pionki.

Gdy wyjdziemy w trakcie rozgrywki (bez zakończenia) przyciskiem „X” u góry okna nasz wynik się nie zapisze do rankingu.

DOKUMENTACJA

Projekt stworzony w języku C++ wykorzystuje bibliotekę graficzna SFML 2.6.
W pliku main() wywołujemy tylko obiekt Menu.
Projekt składa się z klas:

- **Button**
  Każdy objekt tej klasy tworzy osobny przycisk. Ustalamy położenie, rozmiar, tekst i kolory jako argumenty konstruktora. Przycisk ma swoje stany potrzebne podczas operowania kolorami: wciśnięty, kursor na przycisku i bezczynności.
  Funkcja isPressed() zwraca czy przycisk jest wciśnięty.
  Funkcja update() w każdej klatce sprawdza w zależności od położenia kursora w jakim stanie znajduje się przycisk.
  Funkcja render() renderuje przycisk w oknie w którym go wywołamy.
  Funkcja setFont() wczytuje czcionki z pliku.

- **Menu**
  Klasę te wywołujemy w main jako rozpoczęcie naszej gry. Klasa ta zawiera składowe jak rozmiar okienka , tło, listę rankingową. Po stworzeniu obiektu wykonuje się konstruktor w którym znajduje się pętla while i oczekuje na działanie użytkownika. Tworzone są przyciski, które są aktualizowane w niej. W zależności od wciśnięcia danego przycisku tworzone są okienka: gry, rankingu lub potwierdzenia wyjścia z programu. Generuje się w zależności od zwracanego przez klasę mode trybu okienko gry. W konstruktorze wykonuje się ładowanie rankingu z pliku tekstowego. Gdy włączymy rozgrywkę okno menu zamyka się.
  Funkcja loadTexture() ładuje plik .png jako tło programu.

- **mode**
  Klasa, która tworzy okienko wyboru trybu.
  Ma takie składowe jak rozmiar okna, teksty, string, pola do wpisywania.
  Okno tworzone jest w konstruktorze i przyjmuje argumenty: rozmiar czcionki i tekst pytania. Domyślnym trybem jest tryb grania z botem w przypadku wybrania trybu multi pojawia się dodatkowe okno do wpisania nazwy drugiego gracza, w przypadku grania z botem pole to jest wypełniane jako BOT. Możemy przemieszczać się pomiędzy polami do wpisywania poprzez kliknięcie LPM. Wpisujemy z klawiatury nazwę i możemy usuwać znak po znaku klawiszem Backspace, używamy w tym celu std::string aby łatwo usuwać i dodawać pop and push jak na stosie.
  Funkcja bool_mode() zrwaca wybór (true – single, false – multi).
  Funkcja seText() ustawia tekst nagłówkowy – kolor, położenie, rozmiar, czcionkę.
  Funkcja setFont() wczytuje czcionkę z pliku.
  Funkcja setInputFilds() ustawia pola pod wpisywanie m.in. czcionkę, rozmiar, położenie, kolor.
  Funkcja get_name() zwraca nazwę pierwszego gracza.
  Funkcja get_name2() zwraca nazwę drugiego gracza.

- **Confirm**
  To klasa która wyświetla małe okienko pytające czy na pewno chcesz wyjść z programu, po potwierdzeniu następuje wyjście do okna poprzedniego.
  Sklada się z napisu nagłówkowego-pytnia i dwóch przycisków: tak i nie.
  W konstruktorze jest pętla która czeka na wybór użytkownika.
  Funkcja isConfirm() zwraca czy nastąpiło potwierdzenie wyjścia.
  Funkcja setText() ustawia tekst nagłówkowy jego kolor, położenie, rozmiar.
  Funkcja setFont() wczytuje czcionki z pliku.

- **Pawn**
  Klasa tworząca pionka z argumentami: numer pionka, kolor, położenie i promień.
  Przeciążamy konstruktor aby przyjmował inne argumenty. Większość składowych ma ustaloną wartość domyślną. Używamy tej klasy aby wyświetlić żółte większe „kropki" które oznaczają możliwe ruchy.
  Przeciążamy operator aby przypisywać pionki.
  Funkcja render() renderuje pionka w wybranym oknie.
  Funkcja update() przez referencję podaje wybranego pionka, którego wybieramy sprawdzając czy kursor jest na pionku.
  Funkcja update_move() aktualizuje i sprawdza czy wybrany został ruch.
  Funkcja setPawnPosition() ustawia pozycje pionka.
  Funkcja getPawn_number zwraca numer pionka.
  Funkcja getPawn_color(0 zwraca kolor pionka.
  Funkcja print_pawn() wyświetla informacje o pionku, używanapodczas debugowania w konsoli.
  Funkcja set_live() ustawia życie pionka, w przypadku gdy ktoś zbije pionka musimy go usunąć.
  Funkcja get_live() zwraca czy pionek żyje.
  Funkcja setPawn_color() ustawia kolor pionka na podany w argumencie.

- **Filds**
  Klasa pomocnicza przechowywująca tablice z koordynatami x i y każdego pola na planszy, tablice zajętości która przechowuje numery pionków na odpowiednim miejscu.
  Funkcja is_not_occupied() zwraca czy zadane miejsce z argumentów jest wolne.
  Funkcja print_position_occupied() to funkcja która wyświetla w konsoli tablice zajętości.

- **Game**
  Klasa tworzy okno, w której następuje cała rozgrywka.
  Klasa posiada wiele zmiennych przechowują: teksturę czcionke, teksty, rozmiary, liczniki, tablice pionków, obiekt filds, vektor z dostępnymi ruchami, wskaźnik na nowe możliwe ruchy które tworzymy dynamicznie w zależności ile mamy możliwości ruchu.
  Do konstruktora przekazujemy wybór trybu i nazwy graczy.
  Funkcja create24pawns() tworzy pionki od 0 do 11 w kolorze zielonym a pozostałe w czerwonym.
  Funkcja setPawnPosition() przypisuje każdemu pionkowi odpowiadające mu miejsce w tablicy zajętości i miejsce na oknie poprzez koordynaty z FIlds.
  Funkcja render_pawns() w pętli renderuje wszystkie pionki.
  Funkcja update_pawns() w pętli aktualizyje pionki jeżeli jeszcze żyją.
  Funkcja render_ptr_pawns() renderuje żółte pionki jako wybory.

Funkcja update_ptr_pawns() aktualizuje wszystkie stworzone dynamicznie możliwości ruchów. W tej funkcji również jeżeli pionek został wybrany usuwa dynamicznie utworzone pionki.
Funkcja possible_transmitions() dodaje do wektora możliwych ruchów te możliwe dla wybranego pionka.
Funkcja bot_move() wykonuje ruch bota poprzez wywołanie analizy i wybraniu najodpowiedzniejszego ruchu.
Funkcja loadTexture() ładuje grafikę na tło.
Funkcja seFont() ładuje czcionkę z pliku.
Funkcja setText() ustawia wszystkie potrzebne teksty na oknie.
Funkcja vector_module() oblicz odległość pomiędzy dwoma wektorami.
Funkcja vector_deleted_move_pawn() zwraca pozycję pionka którego trzeba usunąć.
Funkcja get_counttime() zwraca czas rozgrywki w sekundach.
Funkcja get_countmove() zwraca liczbę ruchów.
Funkcja check_win() sprawdza ile pionków każdego gracza zostało zbitych i jeżeli liczba ta równa się 12 to gra kończy się i wygrywa przeciwnik.

- **Bot**
  Klasa zawiera składowe zawierające wybrane pionki i ruchy i również wartości danego ruchu.
  Funkcja analyze_board_modified() przyjmuje kopie tablicy zajętości i analizuje ją pod względem możliwych ruchów przeszukując iteracyjnie.
  Funkcja value_of_move() analizuje wybrany każdy ruch pod względem tego jaką ma wartość, możliwość zbicia ma najwyższą wartość, ruchy które prowadzą do rogów lub zbliżają do siebie pionki mają wyższe wartości.
  Funkcja set_max() sprawdza czy nie ruch się nie powtarza i zmienia wybór bota jeżeli znajdzie lepszy wartościowo ruch.
  Funkcja get_move() zwraca wybrany ruch przez bota.
  Funkcja get_selected() zwraca wybranego przez bota pionka.
  Funkcja reset() resetuje wybory bota.

- **Record**
  To klasa przechowywująca dane o każdej rozgrywce, nazwy graczy i liczniki.
  Konstruktor poprzez listę inicjalizującą przypisuje dane do zmiennych.
  Funkcja get_move() zwraca liczbę ruchów.
  Funkcja get_time wraca czas w sekundach.
  Funkcja get_p1() zwraca nazwę pierwszego gracza.
  Funkcja get_p2() zwraca nazwę drugiego gracza.
  Tworzymy funkcje zaprzyjaźnione przeciążenia operatorów aby wczytywać z pliku i wpisywać do pliku.

- **RankingList**
  To klasa która przechowuje listę rekordów w vektorze.
  Funkcja add() dodaje rekord do listy od razu ją pozycjonując, wg licznika ruchów a później według czasu.
  Funkcja loadFile() ładuje z pliku zapisaną wcześniej listę.
  Funkcja saveFile() zapisuje aktualną listę do pliku.
  Funkcja display() wyświetla w konsoli listę ładnie ją listując.
  Zaprzyjaźniona funkcja przeciążenia operatora pozwala zapisać całą listę do pliku.

- **DisplayRanking**
  Klasa wyświetla listę rankingową.
  Przekazywana przez referencję do konstruktora zostaje wyświetlona w oknie.
  Funkcja setFont() ładuje z pliku czcionkę.
  Funkcja setText() ustawia tekst i jego atrybuty.

- **TheEnd**
  Klasa wyświetla okno po wygraniu lub przegraniu.
  Konstruktor przyjmuje argumenty: nazwę wygranego, liczbę ruchów, czas.
  Wyświetla je na ekranie podkreślając wygranego.
  Funkcja setFont() ładuje czcionkę z pliku.