AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Gra Tetris

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja 3 rok

Mariusz Więcławek & Andrzej Filipowski

Grupa: Piątek 14:40

prowadzący: Rafał Frączek

30.12.2021

1. Opis projektu

Jako projekt wybraliśmy temat nr 59 z listy proponowanych tematów tj. „Design and implement classes for a tetris game”. A więc zajęliśmy się wykonaniem popularnej gry tetris, w której to gracz za pomocą strzałek steruje losowo generującymi się figurami cyklicznie zmierzającymi ku dołowi planszy. Gracz ustawia figury tak aby pasowały do siebie i wypełniały plansze bez powstawania luk między nimi. Gdy gracz zapełni całość pól poziomo zdobywa punkt, pozioma linia tych klocków znika, a pozostałe figury przesuwają się o jedno pole w dół. Gra kończy się w momencie gdy figury zetkną się ze szczytem obszaru gry.

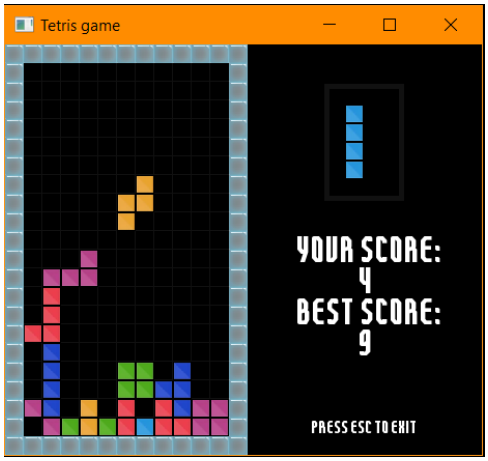
Główna funkcjonalność gry tetris została zaimplementowana w klasie Tetris. Zawiera ona niezbędne pola oraz metody zapewniające poprawne działanie głównego programu.

2. Project description

As a project, we chose the theme No. 59 from the list of proposed topics, ie Design and implement classes for a tetris game. So, we started to make a popular tetris game, in which the player uses arrows to control randomly generated figures cyclically moving towards the bottom of the board. The player arranges the pieces so that they fit together and fill the boards without creating gaps between them. When the player fills all the squares horizontally, he scores a point, the horizontal line of these blocks disappears, and the remaining pieces move down one square. The game ends when the pieces touch the top of the play area.

The main functionality of the tetris game has been implemented in the Tetris class. It contains the necessary fields and methods to ensure correct operation of the main program.

3. Instrukcja użytkownika



Gdy uruchomimy program rozpoczyna się gra. Program generuje losowe figury (złożone z 4 klocków/kwadratów) w losowym kolorze. Sterujemy nimi za pomocą strzałek lewo-prawo. Gdy chcemy wykonać obrót daną figurą klikamy strzałkę w górę, obrót możemy wykonać tylko wtedy gdy jest to możliwe tj. obrót figury nie wyjdzie poza obszar gry oraz nie zetknie się z inną figurą. Jeśli chcemy aby figura opadała szybciej trzymamy strzałkę w dół. Staramy się ustawiać figury tak aby zapełniały wszystkie pola bez powstawania przerw między klockami. Gracz może sprawdzać następną wylosowaną figurę po prawej stronie okna gry. Gdy zapełnimy całą linię klocków poziomo, linia ta znika i zdobywamy punkt. Pozostałe figury przesuwają się o jedno pole w dół. Gdy figury zostaną ułożone pod sam szczyt kończy się gra oraz zapisywany najlepszy wynik. W kolejnych rundach gracz może pobijać swoje rekordowe wyniki. Po przegranej rundzie aby rozpocząć nową klikamy klawisz spacji, co jest sygnalizowane po prawej stronie okna gry. Aby zakończyć działanie programu klikamy klawisz Escape.

4. Kompilacja

Jako iż korzystamy z dodatkowej biblioteki SFML 2.5.1 niezbędna jest odpowiednia konfiguracja kompilatora w swoim środowisku programistycznym. Informacje jak to zrobić znajdziemy na stronie producenta biblioteki SFML. Niezbędne linki:

Konfiguracja na VS: https://www.sfml-dev.org/tutorials/2.5/start-vc.php

SFML na systemach LINUX: https://www.sfml-dev.org/tutorials/2.5/start-linux.php

5. Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

- *Tetris.hpp*, *Tetris.cpp* – deklaracja oraz implementacja klasy `Tetris`.

6. Zależności

W projekcie wykorzystano następujące dodatkowe biblioteki:

- SFML 2.5.1 (ang. Simple and Fast Multimedia Library) - wieloplatformowa biblioteka programistyczna ułatwiająca tworzenie gier oraz programów multimedialnych.

  Strona internetowa: https://www.sfml-dev.org/index.php

7. Opis klas

W projekcie utworzono następujące klasy:

- *Tetris.cpp* – reprezentuje niezbędną funkcjonalność gry tetris
  - *Pola:*
    - **int m_color** – zmienna typu int przechowująca wylosowany kolor figury,
    - **int m_next_color** – zmienna typu int przechowująca wylosowany kolor następnej figury,
    - **int m_fig** – zmienna typu int przechowująca wylosowaną figurę,
    - **int m_game_area[X][Y]** – tablica dwuwymiarowa typu int przechowująca cały obszar gry, aktualne figury znajdujące się na nim.
    - **Point m_Point_position[4]** – tablica struktur Point (struktura Point zawiera zmienne x,y typu int), tablica przechowuje cztery aktualne współrzędne x,y figury na obszarze gry (cztery współrzędne ponieważ figura składa się z 4 klocków).
    - **Point m_Point_pos_temp[4]** – zmienna tymczasowa do której kopiujemy dane z tablicy struktur Point_Position[4],
    - **Point m_Point_next_figure[4]** – określa współrzędne punktu następnej figury która pojawi się na obszarze gry,
    - **Point m_Centre** – struktura Point zawierająca współrzędne punktu który jest środkiem figury, wykorzystywana w celu obrotu figury,
    - **int m_figures[7][4]** – dwuwymiarowa tablicy typu int która zawiera współrzędne 7 różniących się figur składających się z 4 punktów.
  - *Konstruktor:*
    - **Tetris(void)** – w konstruktorze tym inicjujemy współrzędne figur, czyli jej kształt (m_figures[7][4]) oraz inicjujemy początkowe wartości dla pierwszej i następnej wylosowanej figury i jej losowego koloru. Pozostałe wartości pól ustawiamy na 0.

- Getter'y i setter'y:
  - **int get_color(void) const** – zwraca wartość zmiennej m_color,
  - **void set_color(int new_color)** – ustawia wartość zmiennej m_color,
  - **int get_next_color(void) const** – zwraca wartość zmiennej m_next_color,
  - **void set_next_color(int new_color)** – ustawia wartość zmiennej m_next_color,
  - **int get_fig(void) const** – zwraca wartość zmiennej m_fig,
  - **void set_fig(int new_fig)** – ustawia wartość zmiennej m_fig,
  - **Point* get_Point_position(void)** - zwraca wartość tablicy struktur m_Point_position[4],
  - **Point* get_Point_next_figure(void)** - zwraca wartość tablicy struktur m_Point_next_position[4],
  - **int** get_game_area(void)** - zwraca wartość tablicy dwuwymiarowej m_game_area[X][Y],

- *Metody:*
  - **void move_position(int x_position)** – metoda przesuwa figurę poziomo (w lewo lub prawo), sterowane za pomocą zmiennej podanej jako argument,
  - **void rotate_figure(void)** – metoda obraca figurę o 90 stopni,
  - **void fast_falling(void)** – medota przyśpiesza opadanie figury,
  - **bool move_check()** – metoda sprawdza czy przesunięcie figury jest możliwe,
  - **bool rotation_check()** – metoda sprawdza czy obrót figury jest możliwy,
  - **bool point_in_free_area()** – metoda sprawdza czy figura znajduje się w wolnym obszarze gry tzn. czy znajduje się w ramce gry i czy nie nachodzi na inne figury,
  - **bool end_game_check()** – metoda sprawdza czy gra się zakończyła,
  - **void clear_game_area(void)** – metoda czyści cały obszar gry z występujących figur
  - **int which_row_is_full()** – metoda sprawdza który wiersz obszaru gry jest pełny,
  - **void clear_row(int row)** – metoda usuwa dany wiersz obszaru gry oraz przesuwa pozostałe figury o jeden wiersz w dół,
  - **void create_figures(void)** – metoda tworzy kolejne losowe figury o losowych kolorach,

8. Zasoby

W projekcie wykorzystywane są następujące pliki zasobów:

- `images\colors.png` – obrazek zawierający wszystkie kolory z których tworzy się figura.
- `images\tetrisbackgroung.png` – obrazek zawierający tło naszej gry.
- `Fonts\FT_BetonBold.otf` – czcionka tekstu który umieszczamy w oknie gry.

9. Dalszy rozwój i ulepszenia

Kolejną funkcjonalnością naszego programu mogłoby być wprowadzenie zwiększania trudności rozgrywki wraz z upływem czasu np. cyklicznie co 1 min układania klocków zwiększamy prędkość opadania klocków.

Następnym pomysłem jest opracowanie innego sposobu punktowania. W naszym przypadku są zliczane tylko ilości zapełnionych wierszy. Ciekawym rozwiązaniem byłoby otrzymywanie punktów za ilość zapełnionych wierszy w danym cyklu, tj. gdy gracz zapełni w jednym ruchu jeden wiersz otrzymuje X punktów, jeśli zapełni dwa wiersze w jednym ruchu to otrzyma tych punktów więcej itd…

Ciekawym rozwiązaniem było by poszerzenie funkcjonalności menu gry. Gdy uruchomimy grę wita nas menu opcji, w którym użytkownik mógłby zmieniać konfiguracje np. tło gry, sterowanie, gra solo/duo itd.

10. Inne

Brak.