Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Projekt: Grafika komputerowa	
Gra Tetris 2D	Autorzy: **Bartosz Wawrzyk** **Jakub Witas**
Data wykonania: **24.01.2022**	Grupa: **3ID11A**

1. Opis oraz funkcjonalność gry

Przygotowana gra Tetris, bazujea na oryginalnej produkcji stworzonej przez rosyjskiego programistę Alexeya Pażitnova oraz jego współpracowników w 1984. Gra rozpoczyna się na prostokątnej (początkowo pustej) planszy o wymiarach 20 wierszy na 10 kolumn zwanej często matriksem lub tetrionem. W trakcie gry na górze ekranu pojawiają się klocki złożone z czterech małych kwadratów określanych jako bloki lub „tetrimino". Gracz ma do dyspozycji 7 rodzajów takich bloków określanych literami alfabetu ze względu na ich kształt (np. długi podłużny blok określamy literą „I"). Gra trwa aż do momentu, w którym klocek nie będzie mógł pojawić się na planszy. Celem gracza jest przesuwanie (poprzez ruchy w prawo, lewo, w dół oraz rotacje) bloków na planszy w taki sposób, aby zapełniły wiersz na całej szerokości matriksa. Następnie gracz otrzymuje stosowną liczbę punktów, „ukończony" wiersz zostaje usunięty, a reszta bloków znajdujących się na planszy opada w dół osiadając na wolnych miejscach poniżej. Gracz podczas rozgrywki może usunąć maksymalnie 4 wiersze. Sytuację taką nazywamy „tetrisem".

Nasza gra została oparta o bibliotekę graficzną SFML w wersji 2.5.1 zawierającej prosty interfejs, oraz wiele przydatnych narzędzi oraz funkcji do tworzenia aplikacji lub gier. Dużą zaletą biblioteki jest obsługa wielu języków programowania, bardzo prosta i przejrzysta dokumentacja z przykładami oraz kompatybilność z popularnymi systemami operacyjnymi takimi jak: Windows, Linux, czy MacOS. Więcej o SFML przeczytać można pod tym linkiem: https://www.sfml-dev.org/.

Środowisko programistyczne, z którego korzystaliśmy podczas tworzenia projektu to Visual Studio 2019, który zapewnia rozbudowany interfejs, oraz wiele przydatnych funkcji i jest jednym z najpopularniejszych rozwiązań wśród programistów.

Gra została oparta na oryginalnej produkcji Tetris dlatego można wymienić i porównać następujące funkcjonalności i mechaniki:

- Rozgrywka odbywa się w oknie 280x400 pikseli (szer. x wys.)
- Plansza jest podzielona na 20 wierszy i 10 kolumn, po których gracz może poruszać blokami
- Podobnie jak w oryginale skorzystaliśmy z 7 rodzajów tetrimino, z których każdy ma przydzielony indywidualny kolor co pozwala odróżnić go od reszty i ułatwić rozgrywkę
- Skorzystaliśmy również z tradycyjnej mechaniki według której bloki na planszy generowane są w sposób losowy
- Rodzaje tetrimino według notacji literowej to:

  a) Tetrimino „I" – cztery elementy w jednym szeregu
  b) Tetrimino „T" – trzy elementy w rzędzie i jeden dołączony do środkowego elementu
  c) Tetrimino „O" – cztery elementy połączone w kwadrat
  d) Tetrimino „L" – trzy elementy w rzędzie i jeden dołączony do lewego elementu od spodu
  e) Tetrimino „J" – trzy elementy w rzędzie i jeden dołączony do prawego elementu od spodu
  f) Tetrimino „S" – tetrimino „O" po przesunięciu dwóch górnych elementów w prawo
  g) Tetrimino „Z" – tetrimino „O" po przesunięciu dwóch górnych elementów w lewo

- Sterowanie odbywa się przy pomocy strzałek kierunkowych tj. Lewo, Prawo, Dół, a obrócić blok możemy strzałką kierunkową Góra. Natomiast po naciśnięciu „spacja" klocek spada na sam dół oraz po wciśnięciu Esc gra się pauzuje
- Wraz z postępem gry klocki samoczynnie, oraz niezależnie od ruchu gracza poruszają się w dół planszy, dlatego na potrzeby zaimplementowania tej mechaniki skorzystaliśmy z tzw. „timer'a", który pozwolił uzyskać pożądany efekt poruszania się o jeden wiersz niżej z opóźnieniem 0.5 sekundy
- Po ukończeniu jednego lub kilku wierszy są one usuwane, a reszta bloków znajdujących się powyżej opada na ich miejsce. Następnie gracz otrzymuje stosowną ilość punktów, a gra toczy się dalej
- Warto również wspomnieć o systemie kolizji, który na bieżąco sprawdza czy dany ruch może zostać wykonany czy też nie. Podstawowe przypadki działania kolizji to np. sprawdzanie czy dany blok znajdujący się obok ściany może zostać obrócony, lub czy klocek opadł już na stos znajdujący się na dole planszy. W naszej grze bloki zostały oparte na macierzach zatem gdy napotkana zostanie kolizja, akcja którą próbuję wykonać program (np. obrót przy ścianie) jest odwracana tzw. „revert move", tak więc macierz zostanie przywrócona do stanu sprzed kolizji
- Gra kończy się w momencie gdy bloki sięgają szczytu planszy, a gracz nie może wykonać już więcej ruchów

2. Informacje na temat klas i metod

Lista plików wchodzących w skład rozwiązania:

Pliki nagłówkowe:

- **drawableHUD.h** - plik nagłówkowy z implementacją klasy obsługującej interfejs
- **drawableTetromino.h** – plik nagłówkowy z implementacją klasy obsługującej zadania związane z blokami
- **EventHandler.h** – plik nagłówkowy z implementacją klasy obsługi zdarzeń
- **GameEngine.h** – plik nagłówkowy z implementacją klasy służącej za tzw. silnik gry
- **shapeStruct.h** – plik nagłówkowy ze strukturą zawierającą macierze kształtów bloków
- **Vars.h** – plik nagłówkowy z deklaracjami stałych i zmiennych ogólnych koniecznych do działania w całym projekcie, a nie znajdujących się bezpośrednio w klasach

Pliki wykonywalne języka C++:

- **drawableHUD.cpp** – plik wykonywalny zawierający ciała metod z klasy obsługującej interfejs
- **drawableTetromino.cpp** – plik wykonywalny zawierający ciała metod klasy obsługującej zachowania bloków
- **EventHandler.cpp** – plik wykonywalny zawierający ciała metod z klasy obsługującej zdarzenia w programie
- **GameEngine.cpp** – plik wykonywalny zawierający ciała metod z klasy służącej za silnik gry
- **main.cpp** – plik z funkcją main obsługujący pętlę główną gry, oraz zawierający wywołanie metod z poszczególnych klas przekazanych do poprzez silnik gry

Pozostałe pliki:

- **Arial.ttf** – czcionka wykorzystana do wyświetlenia napisów składających się na interfejs
- **background.png** – bitmapa z tłem gry
- **tiles.png** – bitmapa z teksturami klocków, z której wycinane są poszczególne kolory na potrzeby tworzenia bloków

Ogólny opis podstawowych funkcjonalności klas i metod użytych w projekcie:

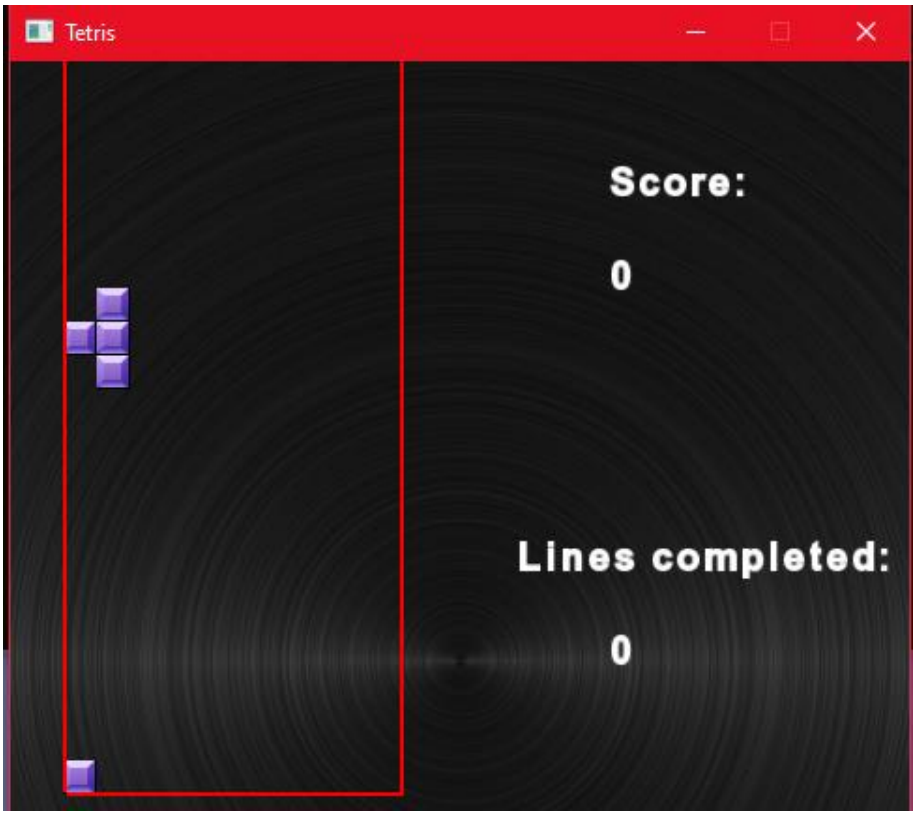
- drawableHUD – klasa działająca jako obsługa interfejsu gry w zakresie takim jak: tło, wizualizacja pola gry, informacje o wyniku i ukończonych wierszach itp. Elementy składowe tej klasy to m.in. referencja do okna głównego w celu uzyskania możliwości rysowania na nim utworzonych obiektów, obiekty klas z biblioteki SFML pozwalające na implementację tekstu, czcionek, czy też tekstur. Metody znajdujące się w tej klasie odpowiadają za tworzenie tła, etykiet tytułowych dla wyniku i ukończonych linii, a także rysowanie linii wyznaczających pole gry.
- drawableTetromino – Rozbudowana klasa odpowiadająca za całą logikę bloku opierającą się na jego konfiguracji, nałożeniu tekstur, koloru, zdefiniowaniu sposobu jego poruszania, usuwaniu wierszy, a także sprawdzaniu kolizji z innymi obiektami gry. Elementy składowe tej klasy to m.in. referencja do okna głównego, utworzenie tekstur i obiektów renderowalnych, a także zmienne niezbędne do konfiguracji bloków takie jak jego rodzaj czy też kolor. Metody tej klasy takie jak np. moveSides, moveDown,

czy Rotate odpowiadają za poruszanie się bloku w obrębie pola gry. Dla samej logiki bloków można natomiast wyróżnić następujące metody: config, pieceFalling, checkCollision, oraz checkLines, które odpowiadają kolejno za przygotowanie struktury bloku o odpowiednim kształcie do jego dalszego konstruowania, samoczynne opadanie bloku zgodnie z timerem, sprawdzanie kolizji bloku z elementami gry, czy też sprawdzenie, które i ile wierszy zostało ukończonych aby je usunąć.

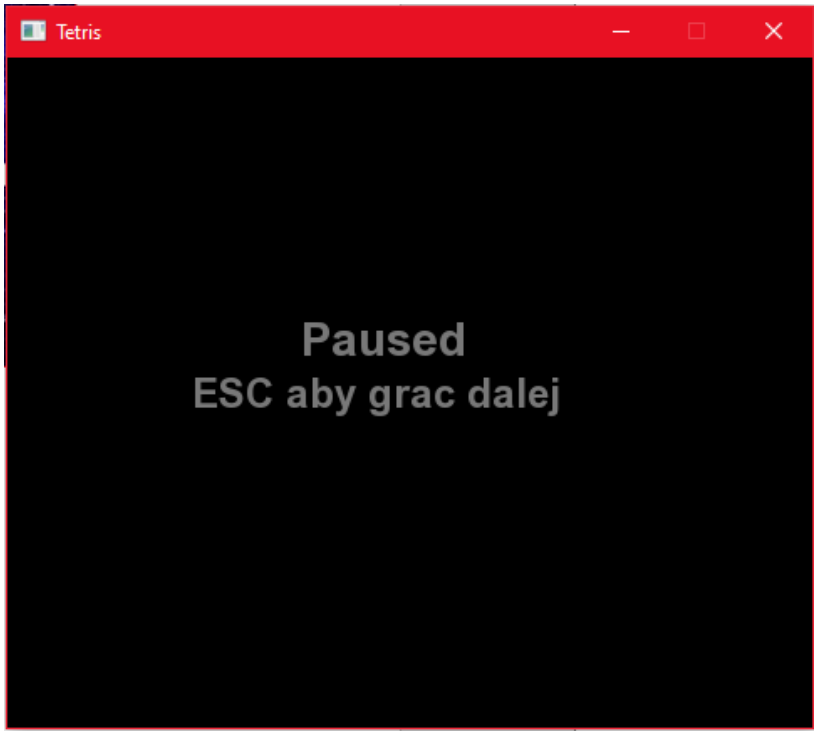
- EventHandler – klasa działająca jako główna obsługa zdarzeń klawiszowych związanych z ruchem bloku w obrębie pola gry. Posiada ona jedną metodę, w której znajduje się przetwarzanie zdarzeń takich jak ruch w płaszczyźnie poziomej i pionowej czy obrót bloku. Dzięki tej klasie po naciśnięciu danego klawisza wykonana zostanie przypisana mu akcja, a ruch bloku będzie widoczny na ekranie.
- GameEngine – główna klasa, którą można nazwać silnikiem gry ponieważ zawiera ona wywołanie wszystkich metod z pozostałych klas, które następnie przekazywane są do funkcji main gdzie znajduje się pętla gry. Tak jak pozostałe klasy zawiera ona referencję do okna głównego w celu uzyskania możliwości rysowania w jego obrębie. Znajdują się tutaj również wskaźniki shared pointer pozwalające na uzyskanie dostępu do metod publicznych pozostałych klas, tak aby móc je wywołać na potrzeby poszczególnych metod z klasy GameEngine. Poszczególne metody tej klasy odpowiadają kolejno za następujące akcje: draw() – główna metoda rysująca wszystkie obiekty w oknie gry, update() – metoda aktualizująca i wykonująca zdarzenia zgodnie z określonym licznikiem czasu, moveEvents() – metoda przekazująca do funkcji main obsługę zdarzeń z klasy EventHandler.
- ShapeStruct – W pliku tym znajdują się deklaracje stałych i zmiennych wykorzystywanych na potrzeby logiki i konfiguracji bloku, oraz pola gry. Struktura Point odpowiada za deklarację tablic, które zawierają pozycję każdego z kwadracików składających się na blok. Wykorzystywane są do konfiguracji i poruszania blokiem. Natomiast tablica figures wykorzystywana jest dla algorytmu konfigurującego blok w celu jego zbudowania od podstaw.
- Vars – Plik ten zawiera deklarację typów wskaźników shared pointer służących do wykorzystywania metod z poszczególnych klas w innych klasach projektu. W pliku tym znajduje się również struktura zawierająca ogólne zmienne i stałe takie jak: zmienne timera, czy też rozmiaru okna, wykorzystywane w różnych metodach i klasach programu.

3. Ekrany

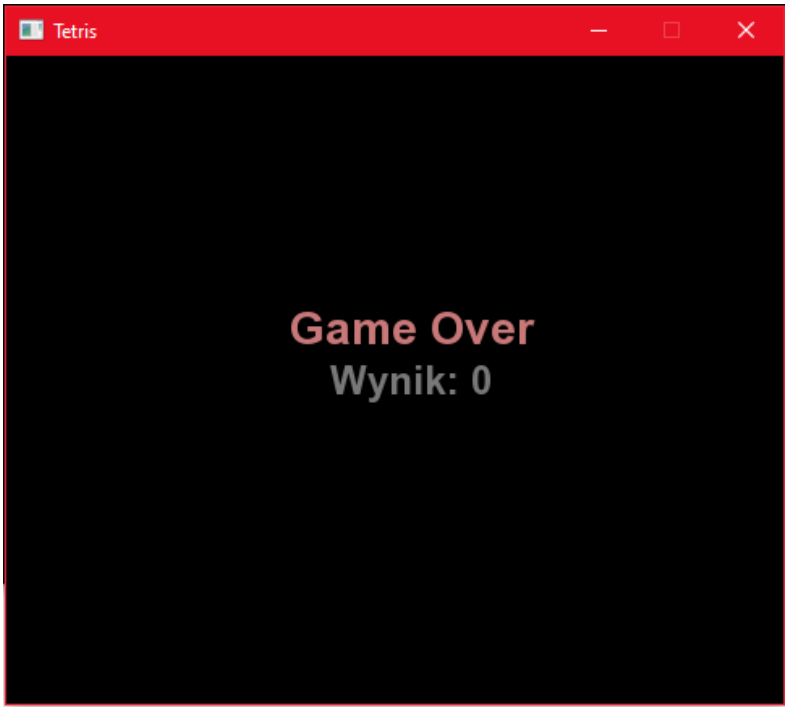
Poniżej znajduje się ekran początkowy, który automatycznie wyświetla grę:



Ekran podczas wciśnięcia przycisku Esc, który pauzuje grę:

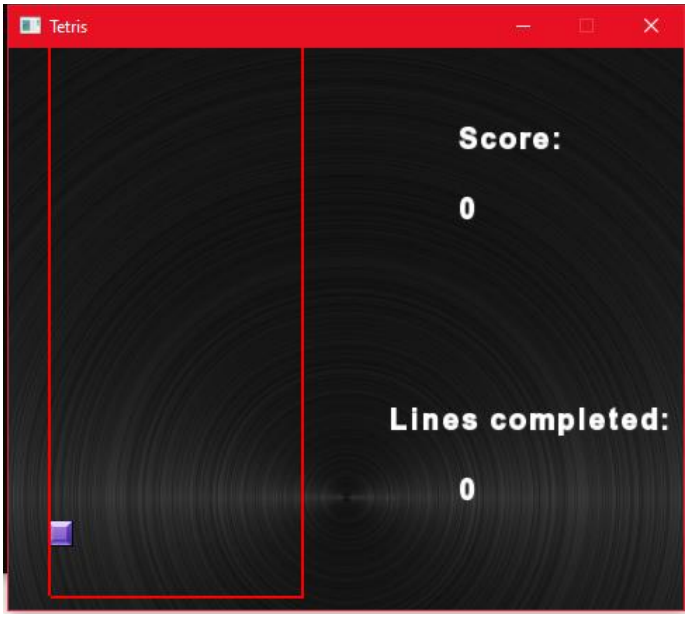


Ekran końcowy, który jest wyświetlany na zakończenie rozgrywki jak klocki dotrą do samej góry i w zależności od wyniku jaki był osiągnięty zostanie on ukazany:



4. Rzeczy, które nie udało się wykonać

- Podczas pracy z grą nie udało się wykonać, aby pierwszy klocek był losowany natomiast podczas każdego uruchomienia gry pierwszy klocek, który się pojawia jest to jeden kwadracik natomiast następne klocki są już losowane.



- Kolejnym elementem, którego nie udało nam się wykonać to brak menu. W momencie uruchomienia programu i wyświetlenia się okienka gra automatycznie startuje. Można jedynie za pauzować jednak, że nie ma opcji aby wybrać, w którym momencie zaczynasz grę oraz skończyć ją przed upływem rozgrywki oraz nie ma opcji, w której można by wybrać spis wszystkich osób, które grały oraz tabelę z wynikami.