Sprawozdanie Projektowe

Przedmiot:

- ***Grafika komputerowa(projekt)***

Temat:

- ***Gra Snake***

Zespół:

- ***Dominik Grudzień***
- ***Patryk Grzywacz***

Informacje:

- ***Rok: 3***
- ***Kierunek: Informatyka***
- ***Grupa: 3ID11B***

Szczegółowy Opis Projektu

Język:

- *Obiektowy C++* ***Technologia:***
- *SFML (Super-Fast Multimedia Library) w wersji 2.5.1*

Biblioteki:

- *Standardowe biblioteki języka C++* ○ *iostream* ○ *fstream* ○ *time*

- *Biblioteka SFML Graphics - do rysowania na ekranie*
- *Biblioteka SFML Audio – do dźwięków i muzyki*

Funkcjonalności:

- *Interaktywne menu gry za pomocą klawiatury i myszki*
- *Dynamiczne poziomy o losowym rozłożeniu przeszkód*
- *Możliwość zapisu i odczytu wyników najlepszych graczy*
- *Klimatyczna muzyka do każdego poziomu*
- *Skalowanie poziomu trudności z każdym poziomem • Możliwość nabijania combo*
- *Bonusy w postaci:*
  - *Złote jabłko dające czasową odporność i więcej punktów*
  - *Dziury oraz Ściany teleportacyjne umożliwiające sprytniejsze poruszanie się po mapie*
  - *Skórka postaci ,która zmienia się w zależności od nabitego wyniku*
- *Możliwość ponownego rozpoczęcia poziomu w zależności od wyboru gracza przy niepowodzeniu*
- *Różnorodne animacje, efekty graficzne oraz dźwiękowe*

Uruchamianie oraz obsługa projektu:

✦ ***Windows***

1. *Poprzez instalator*
    - *Uruchom Setup.exe*
    - *Postępuj dalej wraz z instrukcjami instalatora*
    - *Uruchom SnakeSFML.exe z uprawnieniami administratora*
    - *Ciesz się grą*
2. *Poprzez MS Visual Studio 2019*
    - *Uruchom z głównego folderu Snake SFML.sln*
    - *W ustawieniach projektu zalinkuj odpowiednie biblioteki SFML*
    - *Skompiluj projekt*
    - *Ciesz się grą*

✦ ***Linux***

1. *Otwórz terminal i wpisz komendę ” sudo apt-get install libsfml-dev „ aby pobrać niezbędne dodatkowe pliki do obsługi biblioteki SFML*
2. *Za pomocą terminalu uruchom plik MakeFile komendą „ ./MakeFile ” będąc w folderze głównym*
3. *Ciesz się grą*

Informacje na temat klas, metod i funkcji:

1. ***Klasa Animacja –*** *klasa* **odpowiadajaca za animacje sprite pokarmow i weza.**
    - *Animacja() – konstruktor obiektu klasy ○ Za parametry przyjmuje teksturę obiektu, liczbę obrazków animacji oraz czas zmiany obrazków.*
        - *Nic nie zwraca.*
        - *Tworzenie obiektu Animacji danego Sprite’a.*
    - *Aktualizuj() – metoda do aktualizacji Animacji ○ Za parametr przyjmuje kolejny numer obrazku animacji ○ Zwraca pierwszy obrazek animacji ○ Jej przeznaczeniem jest odpowiednie zmienianie tekstur*
2. ***Klasa Aplikacja –*** *klasa odpowiedzialna za włączanie gry*

- *Aplikacja() – konstruktor obiektu klasy*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*
  - *Tworzy obiekt klasy Aplikacja i tworzy okno główne gry o określonych parametrach*
- *Start () – metoda ta rozdziela Gre na 4 podprogramy*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*
  - *Tworzy odpowiednie obiekty pozmiów i menu oraz wywołuje dalsze funkcje gry*

3. *Klasa Gra – klasa główna Gry odpowiedzialna za działanie silnika gry*
   - *Gra() – konstruktor obiektu klasy Gra*
     - *Za parametry przyjmuje ilość przeszkód na poziomie, numer poziomu oraz warunek ukończenia poziomu.*
     - *Nic nie zwraca*
     - *Inicjalizuje , tworzy i ustawia zmienne oraz obiekty i wywołuje odpowiednie funkcje*
   - *~Gra() – dekonstruktor obiektu klasy Gra*
     - *Nie przyjmuje parametrów*
     - *Nic nie zwraca*
     - *Jego zadaniem jest usunąć wszystkie dynamicznie utworzone tablice*
   - *aktualizujStanGry() - Metoda ta aktualizuje wynik zjedzonych jabłek na ekranie.*
     - *Za parametr przyjmuje aktualny wynik zdobyty przez gracz*
     - *Nic nie zwraca*
     - *Aktualizuje pokazywany na ekranie warunek przejścia do kolejnego poziomu*
   - *gameOver() – metoda do wyboru sytuacji gdy gracz przegra*
     - *Za parametry przyjmuje okno główne aplikacji oraz obiekt gracza i pokarmu*
     - *Metoda zwraca 0 gdy gracz będzie chciał ponowić poziom , a 1 gdy gracz będzie chciał zakończyć granie*
   - *pauza() – metoda do pauzowania gry*
     - *Za parametry przyjmuje okno główne aplikacji oraz obiekt gracza i pokarmu*
     - *Metoda zwraca 0 lub 1 w zależności od wyboru gracz*

- *pauzaPrzedGra() – metoda do ustawiania aktywnej pauzy przed rozpoczęciem poziomu ○ Za parametry przyjmuje okno główne aplikacji oraz*
  obiekt gracza i pokarmu ○
  Nic nie zwraca
- *przegrana() – metoda sprawdza czy gracz przegrał w danym poziomie ○ Za parametry przyjmuje obiekt gracza oraz zegar do sprawdzania czy ochrona nadal trwa*
  - *Zwraca true gdy gracz nie przegrał , a false gdy gracz zjadł samego siebie lub wszedł w przeszkodę*
- *rysujPlansze() – metoda do rysowania planszy i jej ramki w oknie gry ○ Za parametr przyjmuje okno główne gry ○ Nic nie zwraca*
- *silnikPoziomu() – to główna metoda klasy Gra w ,której wywoływane są odpowiednie metody oraz tworzone obiekty ○ Za parametry przyjmuje okno główne gry oraz numer aktualnego poziomu*
  - *Zwraca numery poziomów (1,2,3) lub 0 gdy nastąpi zakończenie gry i wyjście z aplikacji*
- *wygrana() – metoda ta sprawdza czy gracz wygrał w danym poziomie ○ Za parametr przyjmuje aktualny wynik nabity przez gracz (ilość jabłek zjedzonych)*
  - *Zwraca true gdy gracz osiągnie odpowiedni wynik a false gdy jeszcze nie lub jest rozgrywany ostatni poziom*

4. *Klasa Gracz - Klasa odpowiedzialna za obsługę gracza.*
   - *gracz() – konstruktor klasy gracz ○ Za parametr przyjmuje numer rozgrywanego poziomu ○ Nic nie zwraca ○ Metoda ta inicjalizuje zmienne , ładuje tekstury i dźwięki oraz tworzy listę dwukierunkową*
   - *dodajElement() - Metoda odpowiedzialna za obsluge listy dwukierunkowej ○ Nie przyjmuje parametrów ○ Nic nie zwraca*
     - *Metoda dodaje nowy element do listy*
   - *obsluguj() - Metoda zwiazana z ruchem gracza.*
     - *Za parametry przyjmuje wskaźniki na tablice dziur teleportacyjnych i przeszkód, a także ich ilości*
     - *Nic nie zwraca*
     - *Zawarte są w niej prywatne metody związane z ruchem oraz sterowaniem gracza ,które cały czas działają*

- *ochronaKolizji() - Metoda odpowiedzialna za dopasowanie tekstur podczas bonusu ochrony przed kolizja.*
    - *Nie przyjmuje parametrów*
    - *Nic nie zwraca*
- *przejdzNaKoniecListy() - Metoda odpowiedzialna za obsługę listy dwukierunkowej ○ Za parametr przyjmuje podwójny wskaźnik na listę na*
  której opiera się gracz
    - *Nic nie zwraca*
    - *Metoda przechodzi na koniec listy*
- *przejdzPrzezDziure() - Metoda odpowiedzialna za przenoszenie węża przez tunel w inne miejsce planszy (druga dziura). ○ Za parametry przyjmuje wskaźnik na tablicę dziur*
  oraz ich liczbę
    - *Nic nie zwraca*
- *przejdzPrzezSciane() – Metoda odpowiedzialna za przenoszenia węża przy przechodzeniu przez ściany ○ Nie przyjmuje parametrów ○ Nic nie zwraca*
- *ruchGracz() - Metoda odpowiadajaca za poruszanie sie węża.*
    - *Za parametry przyjmuje wskaźniki na tablice dziur teleportacyjnych i przeszkód, a także ich ilości*
    - *Nic nie zwraca ○ Kontroluje wszystkie ruchy węża np.:*
        - *Przejście przez ścianę*
        - *Przejście przez dziurę*
        - *Wejście w przeszkodę*
- *rysuj() – Metoda rysująca elementy gracza ○ Za parametr przyjmuje okno główne gry ○ Nic nie zwraca*
- *samoUkaszenie() – Metoda sprawdzająca czy gracz nie zjadł sam siebie ○ Nie przyjmuje parametrów ○ Zwraca true gdy wąż zje sam siebie, a false gdy przecwinie*
- *sterowanie() - Metoda odpowiedzialna za odczytywanie klawiszy strzałek w celu ustaleniu kierunku poruszania się gracza.*
    - *Nie przyjmuje parametrów ○ Nic nie zwraca*
- *ustawNowaTeksture() - Ustawia nowa tekstura odpowiednio do uzyskanego wyniku. ○ Nie przyjmuje parametrów ○ Nic nie zwraca*

- *ustawTeksture100() - Metoda, która ustawia teksturę nieprzezroczystą po bonusie ochrony przed kolizja.*
    - *Nie przyjmuje parametrów*
    - *Nic nie zwraca*
- *walnijPrzeszkode() - Metoda sprawdzająca czy gracz znajduje się w przeszkodzie. ○ Za parametry przyjmuje wskaźnik na tablicę przeszkód i ich liczbę*
    - *Zwraca true gdy walnie w przeszkodę ,a false gdy przeciwnie*
- *zerujAnimacje() - Metoda odpowiedzialna za ustawienie animacji na start. ○ Nie przyjmuje parametrów ○ Nic nie zwraca*

5. *Klasa Koniec - Klasa do wprowadzania nicku gracza oraz manipulacji listą wyników.*
    - *Koniec() – Konstruktor obiektu klasy Koniec*
        - *Nie przyjmuje parametrów ○ Nic nie zwraca*
        - *Ustawia zmienna wskaźnikową wsk_wyniki na NULL*
    - *wpiszNick() - Metoda ta odpowiada za umożliwienie graczowi wpisanie swojego nicku/pseudonimu.*
        - *Za parametry przyjmuje okno główne gry , typ czcionki oraz tło aktualnego poziomu*
        - *Zwraca pseudonim/nick gracz*
    - *wynikiTXT() - Metoda ta umożliwia znalezienie odpowiedniego miejsca w liście wyników.*
        - *Za parametr przyjmuje nick gracz wpisany w wpiszNick()*
        - *Nic nie zwraca*
        - *Metoda wpisuje 5 najlepszych wyników do pliku wynikowego gry.*
6. *Klasa Menu - Klasa ta odpowiada za dzialanie menu glownego gry.*
    - *Menu() - Jest to konstruktor obiektu klasy Menu ○ Nie przyjmuje parametrów ○ Nic nie zwraca ○ Inicjalizuje on zmienne, obiekty ,tablice itd.*
    - *~Menu() - Jest to dekonstruktor obiektu klasy Menu ○ Nie przyjmuje parametrów ○ Nic nie zwraca ○ Niszczy on dynamicznie utworzone tablice i zmienne*

- *aktualizacjaMenu() - Metoda ta ma za zadanie odpowiednio aktualizować widok wyboru użytkownika.*
    - *Nie przyjmuje parametrów*
    - *Nic nie zwraca*
- *enter() - Metoda ta ma za zadanie zmieniać podstrony menu lub uruchamiać poziom czy tez kończyć działanie aplikacji.*
    - *Za parametr przyjmuje okno główne gry*
    - *Nic nie zwraca*
- *klikMyszka() - Metoda ta oblicza czy myszka najechała na któryś interaktywny element przy kliknięciu.*
    - *Za parametry przyjmuje pozycje x oraz y myszki w momencie kliknięcia myszką, w odniesieniu do lewego górnego rogu okna gry oraz okno główne gry*
    - *Nic nie zwraca*
- *poruszajTlo() - Metoda ta ma za zadanie poruszać tłem menu głównego.*
    - *Nie przyjmuje parametrów*
    - *Nic nie zwraca*
- *przygotujStrone() - Metoda ta przygotowuje odpowiednie zasoby do rysowania dla danej podstrony i wyboru użytkownika.*
    - *Nie przyjmuje parametrów*
    - *Nic nie zwraca*
- *ruchMyszka() - Metoda ta oblicza czy myszka najechała na któryś interaktywny element w trakcie ruchu.*
    - *Za parametry przyjmuje pozycje x oraz y myszki w momencie ruchu, w odniesieniu do lewego górnego rogu okna*
    - *Nic nie zwraca*
- *rysuj() - Metoda ta ma za zadanie rysowanie odpowiedniego podmenu menu głównego .* ○ *Za parametr przyjmuje okno główne gry* ○ *Nic nie zwraca*
- *start() – Jest to metoda która uruchamia menu główne* ○ *Za parametr przyjmuje okno główne gry* ○ *Zwraca 1 gdy użytkownik chce rozpocząć grę, a 0 gdy chce zakończyć działanie aplikacji*

7. *Klasa Pokarm - Klasa odpowiedzialna za obsługę pokarmu dla węża.*
    - *Pokarm() – jest to konstruktor obiektu klasy Pokarm* ○ *Za parametry przyjmuje ścieżki do pliku z teksturą pokarmu oraz jej animacji a także obiekt klasy*

Vector2u z wartościami do tworzenia animacji pokarmu
- *Nic nie zwraca*
- *Inicjalizuje wszystkie potrzebne zmienne, ładuje tekstury i je przypisuje do sprite'ów oraz ustawia animacje*

- *~Pokarm() - jest to dekonstruktor obiektu klasy Pokarm*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*
  - *Usuwa efekt oraz animacje pokarmu*
- *rysuj() - Metoda odpowiedzialna za rysowanie animacji pokarmu oraz efektu pojawiania się go na planszy.*
  - *Za parametr przyjmuje okno główne aplikacji*
  - *Nic nie zwraca*
- *sprawdzCzyZjedzony() - Metoda sprawdzająca czy gracz wszedł w kolizje z pokarmem.*
  - *Za parametr przyjmuje obiekt gracza*
  - *Zwraca true gdy gracz wszedł w kolizje, false gdy przeciwnie*
  - *Jeśli gracz wszedł w kolizję to ustalane jest kolejne położenie pokarmu. Brane są pod uwagę przeszkody oraz ciało węża.*
- *ustawPokarm() - Metoda poszukująca nowego położenia dla pokarmu.*
  - *Za parametry przyjmuje obiekt gracza, podwójny wskaźnik na dynamiczną tablicę planszy , wskaźnik na tablicę przeszkód oraz ich liczbę*
  - *Nic nie zwraca*
- *wyzerujAnimacje() - Metoda odpowiedzialna za ustawienie animacji na start.*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*

8. *Klasa Poziom1,Poziom2,Poziom3 – Te klasy mają za zadanie tworzyć odpowiednie wersje obiektu klasy Gra i uruchamiać silnik gry*

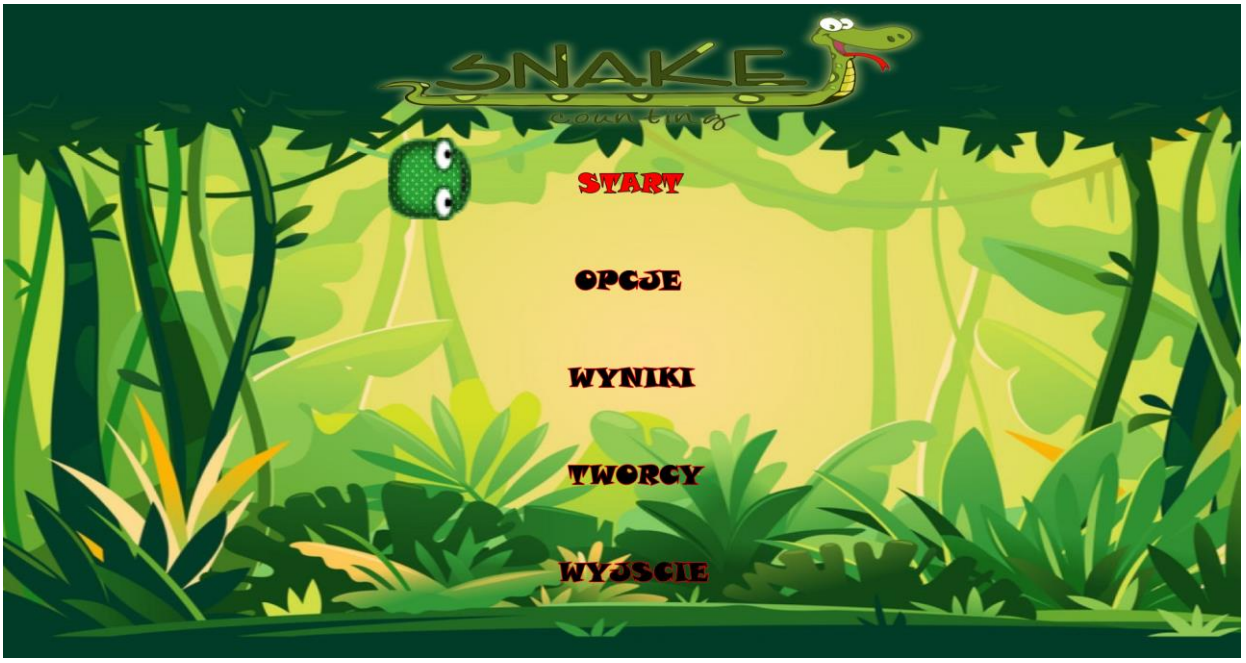
Przykładowo metody dla klasy Poziom1 (pozostałe klasy mają takie same metody co klasa Poziom1)

- *Poziom1() – Jest to konstruktor obiektu klasy Poziom1*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*
  - *Ten konstruktor ładuje odpowiednie tło poziomu oraz tworzy obiekt Gra wg. odpowiednich parametrów.*
- *~Poziom1() - Jest to dekonstruktor obiektu klasy Poziom1*
  - *Nie przyjmuje parametrów*
  - *Nic nie zwraca*

- *start() - Ta metoda uruchamia silnikPoziomu() obietku klasy Gra ○ Za parametr przyjmuje okno główne gry ○ Zwraca wartość równą jej poziomowi lub większą jeśli istnieje albo 0 gdy aplikacja kończy swoje działanie poprzez użytkownika*

9. *Klasa Punkty - Klasa ta ma za zadanie przechowywać wynik oraz kombo i je animować.*
   - *Punkty() – Jest to konstruktor obiektu klasy Punkty*
     - *Nie przyjmuje parametrów ○ Nic nie zwraca*
     - *Jego zadaniem jest inicjalizowanie zmiennych oraz pozycji obiektów.*
   - *~Punkty() - Jest to dekonstruktor obiektu klasy Punkty*
     - *Nie przyjmuje parametrów ○ Nic nie zwraca*
   - *animujKombo() – Ta metoda ma za zadanie animować wyświetlane kombo.*
     - *Za parametr przyjmuje 0 lub 1, które ustala czy animacja ma trwać dalej*
     - *Nic nie zwraca*
   - *animujWynik() – Ta metoda ma za zadanie animować wyświetlany wynik ○ Za parametr przyjmuje 0 lub 1, które ustala czy animacja ma trwać dalej*
     - *Nic nie zwraca*
   - *dodajPunkty() - Ta metoda ma za zadanie dodanie punktów do wyniku.*
     - *Za parametr przyjmuje wartość do dodania ○ Nic nie zwraca*
     - *Metoda mnoży otrzymaną wartość i aktualne kombo i dodaje ją do wyniku gracza*
   - *sprawdzWynik() - Ta metoda ma za zadanie sprawdzić aktualny wynik i go zwrócić. ○ Nie przyjmuje parametrów ○ Zwraca aktualny wynik gracza*
   - *zmienKombo() - Ta metoda ma za zadanie dodanie kombo*
     - *Za parametr przyjmuje wartość do dodania do aktualnego kombo*
     - *Nic nie zwraca*

Przykładowe zrzuty ekranu:

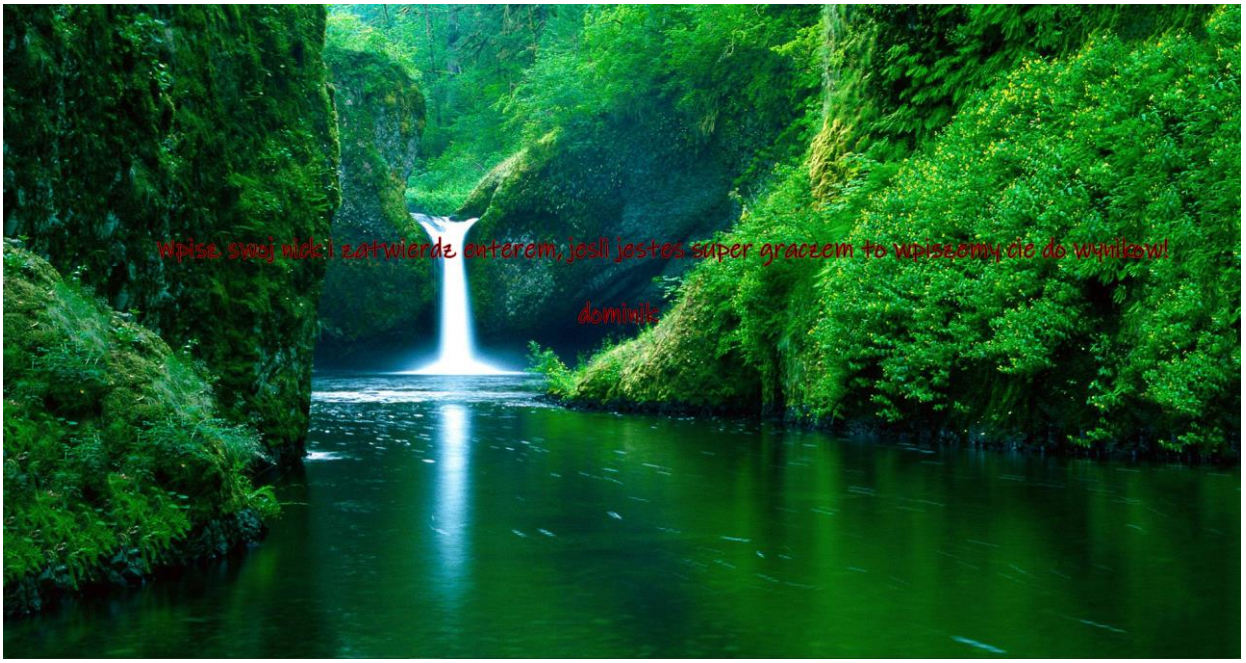


Menu główne gry snake

Poziom 1 gry snake

Poziom 2 gry snake

Poziom 3 gry snake



Zapisywanie punktów zebranych podczas gry do pliku wyniki.txt

Informacje na temat ilości pracy włożonej przez poszczególnych członków zespołu:

1. *Dominik Grudzień :*

- *Mechaniki związane z graczem , pokarmem oraz tworzeniem animacji.*
- *Testowanie i debugowanie aplikacji.*
- *Tworzenie dokumentacji oraz plików MakeFile.*
- *Tworzenie tekstur , grafik oraz muzyki.*

2. *Patryk Grzywacz :*

- *Działanie menu głównego.*
- *Mechaniki związane z poziomami oraz silnikiem gry , punktacji oraz wpisywań wyników.*
- *Tworzenie dokumentacji oraz instalatora gry.*

Wszyscy członkowie zespołu nawzajem sobie podpowiadali przy pisaniu kodu , optymalizowali jego działanie oraz ulepszali mechaniki gry.