Dokumentacja Projektu II – konsola

Temat: Gra “Snake”

Komunikacja Człowiek-Komputer

Wykonujący projekt:
Gabriel Czajkowski

Studia dzienne
Kierunek: Informatyka
Semestr: V
Grupa zajęciowa: PS2

Prowadzący:
mgr inż. Aleksander Sawicki

1. OPIS PROJEKTU

Głównym celem oraz założeniem projektu było zaimplementowanie gry wzorowanej na kultowym Snake'u działającej w trybie tekstowym. Miała ona być przyjazna i atrakcyjna użytkownikowi końcowemu poprzez dodanie animacji oraz wygodnego i intuicyjnego w obsłudze interfejsu. Napisana została w języku C#, w środowisku Microsoft Visual Studio.

Snake jest to gra jednoosobowa, w której gracz kontroluje węża, poruszającego się po obramowanej planszy oraz zjadającego owoce. Pożywienie pojawia się w losowym miejscu. W grze są dwa jego rodzaje: podstawowe oraz specjalne. Za zjedzenie pierwszego z nich gracz otrzymuje ilość punktów zależną od poziomu trudności, długość węża zwiększa się o jeden człon oraz wzrasta jego prędkość. Specjalne owoce pojawiają się niezależnie od podstawowych. Dają one znacznie więcej punktów, jednak w zamian za to, długość węża zwiększa się o dwa człony oraz znacznie wzrasta jego prędkość. Gra kończy się, gdy wąż uderzy głową w ścianę lub część własnego ciała. Gracz kontroluje kierunek ruchu gada za pomocą strzałek (góra, dół, lewo, prawo) lub klawiszy WSAD (W – góra, S – dół, A – lewo, D – prawo).

2. OPIS FUNKCJONALNOŚCI

- Po uruchomieniu programu zostanie wyświetlona animacja logo gry "SNAKE".
- Możliwość wyboru jednego z trzech poziomów trudności:
    - łatwy,
    - średni,
    - trudny
- W zależności od wybranego poziomu różnić się będzie prędkość początkowa węża, ilość zdobywanych punktów oraz zwiększenie jego szybkości za każdy zjedzony owoc.
- Możliwość wyboru własnych ustawień dotyczących:
    - prędkości węża (od powolnej aż po bardzo szybką),
    - częstotliwości pojawiania się specjalnych owoców (od małej aż po bardzo dużą).
- Po skorzystaniu z jakiejkolwiek z dwóch powyższych opcji personalizacji ustawień, poziom trudności zostanie ustawiony na "OWN", a wąż nie będzie przyspieszał po zjedzeniu owocu.
- W momencie uruchomienia aplikacji będzie słychać muzykę uprzyjemniającą wrażenia z gry.
- Możliwość wyłączenia/włączenia utworu lub jego zmiana na inny.
- Zapis i odczyt wyników do/z pliku txt.
- Możliwość wyświetlenia 10 najlepszych wyników wraz z nazwą gracza, miejscem które zajął, liczbą uzyskanych punktów, datą, oraz poziomem trudności (EASY/MEDIUM/HARD/OWN).
- Możliwość zobaczenia autora aplikacji.
- Konieczność wprowadzenia odpowiedniej nazwy gracza przed rozpoczęciem gry (od 3 do 12 znaków).
- Animacja odliczania do rozpoczęcia rozgrywki.
- Podczas gry wyświetlana nazwa gracza, poziom trudności na którym gra, ilość zdobytych przez niego punktów, standardowy oraz animowany specjalny owoc.
- Po porażce wyświetlanie odpowiedniego napisu z wykorzystaniem ASCII art oraz dodatkowej informacji w przypadku osiągnięcia nowego rekordu.
- Możliwość łatwego rozpoczęcia kolejnej gry po przegranej.
- Sterowanie wężem za pomocą strzałek (góra, dół, lewo, prawo) oraz klawiszy WSAD (W – góra, S – dół, A – lewo, D – prawo).
- Możliwość wyjścia z gry. Podczas wychodzenia zapytanie o to, czy jest się pewnym, że chce się wyjść.

3. SZCZEGÓLNIE INTERESUJĄCE ZAGADNIENIA PROJEKTOWE

Najtrudniejszym wyzwaniem okazała się animacja specjalnego owocu. Zrealizowana została poprzez zastosowanie wielowątkowości. Stworzono nowy wątek, który zmienia kolor oraz kształt pożywienia przy pomocy metody Write() klasy Point(). Jest ona wywoływana co jakiś czas, co wizualizuje animacje.

```
#region EXTRA FRUIT ANIMATION
public void FruitAnimation(ref CancellationTokenSource token)
{
    int i;
    while (true)
    {
        lock (listManipulator)
        {
            if (fruitList.Count != 0)
            {
                i = rand.Next(0, fruitList.Count);
                lock (writer)
                {
                    if (token.IsCancellationRequested) break;
                    fruitList[i].Write();
                }
            }
        }

        //FLASHING EXTRA FRUIT
        Thread.Sleep(300);
    }
}
#endregion EXTRA FRUIT ANIMATION
```

Obraz 1. Metoda FruitAnimation() wywołująca metodę Write() klasy Point().

```
private class Point
{
    public int x;
    public int y;
    private bool red = true;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public void Write()
    {
        Console.SetCursorPosition(x, y);
        if (red)
        {
            Console.ForegroundColor = ConsoleColor.Magenta;
            Console.Write("O");
            red = false;
        }
        else
        {
            Console.ForegroundColor = ConsoleColor.DarkMagenta;
            Console.Write("o");
            red = true;
        }
    }
}
```

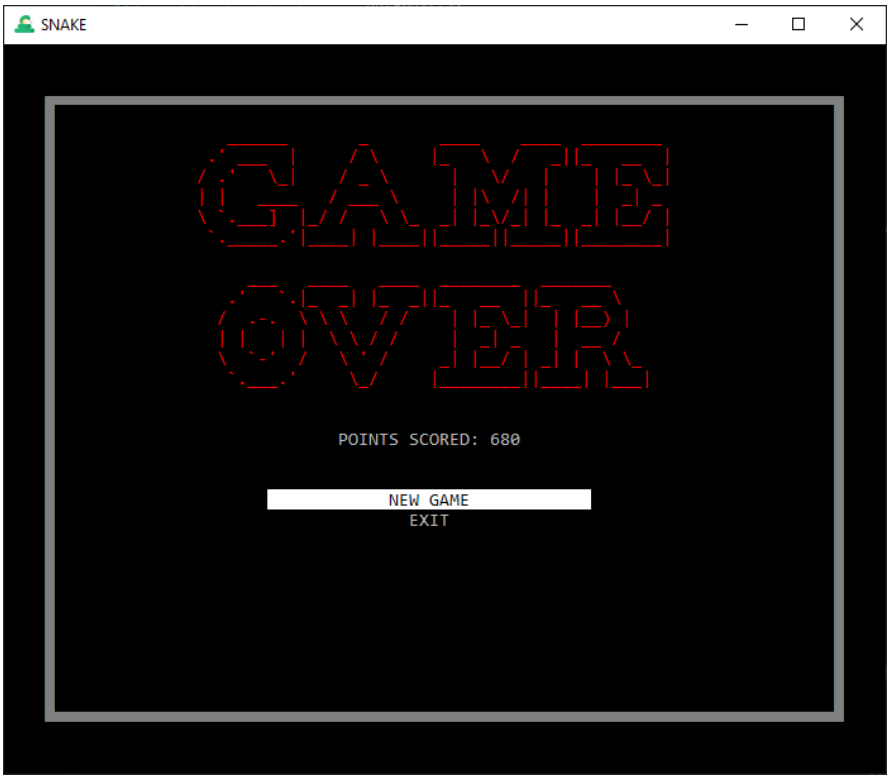
Obraz 2. Metoda Write() w klasie Point() odpowiedzialna za animacje specjalnego owocu.

Kolejnym problemem okazała się animacja logo przy pierwszym wejściu do gry. Za jej realizację odpowiada metoda ShowAnimatedLogo() w której znajduje się pętla for() wyświetlająca poszczególne linijki nagłówka oraz zatrzymująca wątek na pewien czas w celu wizualizacji animacji.

```
private void ShowAnimatedLogo()
{
    ANIMATION 1

    #region ANIMATION 2
    int x1 = startDravingHeaderXPosition;
    int y1 = startDravingHeaderYPosition;
    int x2 = x1;
    int y2 = 4;
    int height = headerList.Count - 1;
    int height2 = 0;

    Console.ForegroundColor = ConsoleColor.Green;
    for (int i = 3; i > 0; i--)
    {
        Console.SetCursorPosition(x1, y1--);
        Console.Write(headerList[height--]);
        Console.SetCursorPosition(x2, y2++);
        Console.Write(headerList[height2++]);
        Thread.Sleep(1000);
    }

    Console.ResetColor();
    #endregion ANIMATION 2
}
```

Obraz 3. Metoda ShowAnimatedLogo() odpowiedzialna za animacje logo.

Ostatnim i najmniejszym problemem było wykorzystanie ASCII art. Zastosowano je do wyświetlania nagłówka, informacji o porażce, odliczania do rozpoczęcia gry oraz imienia i nazwiska autora.



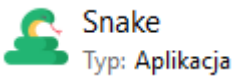
Obraz 4. Przykład wykorzystania ASCII art w momencie porażki.

```
string[] game = File.ReadAllLines("ASCII Arts/Game over/Game.txt");
string[] over = File.ReadAllLines("ASCII Arts/Game over/Over.txt");

Console.ForegroundColor = ConsoleColor.Red;
for (int i = 0; i < game.Length; i++)
{
    Console.SetCursorPosition(MiddleWindowWidth - (game[0].Length / 2), MiddleWindowHeight - 14 + i);
    Console.Write(game[i]);
}

for (int i = 0; i < over.Length; i++)
{
    Console.SetCursorPosition(MiddleWindowWidth - (over[0].Length / 2), MiddleWindowHeight - 7 + i);
    Console.Write(over[i]);
}
```

Obraz 5. Przykładowy kod obsługujący wyświetlanie ASCII art.

4. INSTRUKCJA INSTALACJI

W celu skompilowania kodu gry oraz uruchomienia aplikacji niezbędne jest posiadanie zainstalowanego środowiska programistycznego obsługującego język C# np. Microsoft Visual Studio. Następnie należy zainicjować plik wykonywalny z rozszerzeniem .exe o nazwie Snake znajdujący się w folderze Debug. Ścieżka: Snake/bin/Debug/Snake.exe



Obraz 6. Ikona do uruchomienia aplikacji.

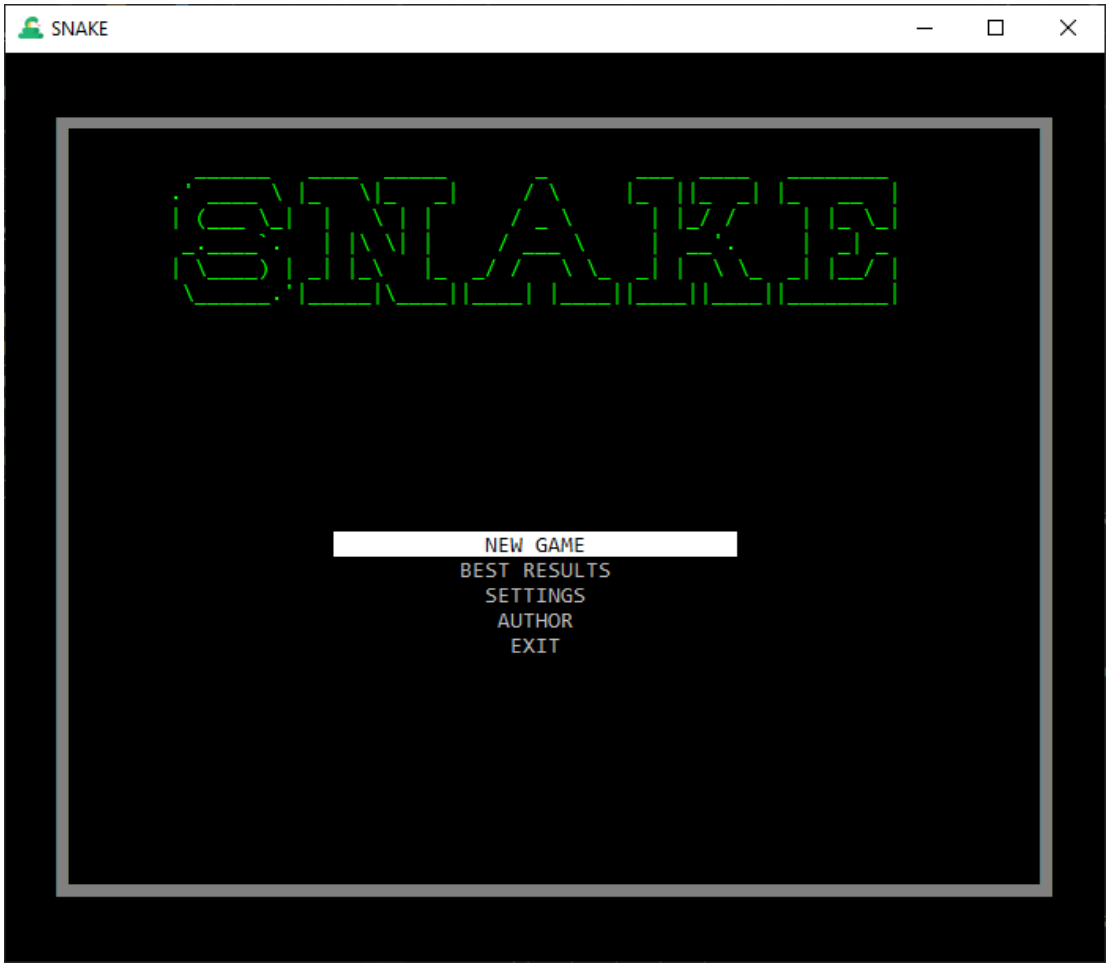
5. INSTRUKCJA KONFIGURACJI

Aplikacja nie ma specjalnych wymagań dotyczących konfiguracji.

6. INSTRUKCJA UŻYTKOWANIA

Po menu można poruszać się za pomocą przycisków:

- strzałka w górę – ruch w górę,
- strzałka w dół – ruch w dół,
- W – ruch w górę,
- S – ruch w dół,
- ENTER – zatwierdzenie zaznaczonej opcji,
- ESC – powrót do poprzedniego widoku lub wyjście z gry.



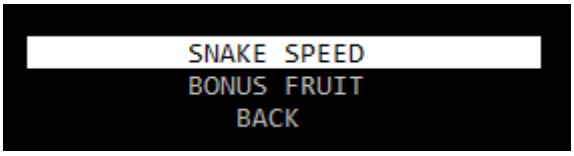
Obraz 7. Widok menu głównego. Aktualnie wybrana pozycja to biały prostokąt z czarnym napisem.

Podczas gry wężem poruszać można za pomocą przycisków:

- strzałka w górę (↑) – zmiana kierunku ruchu na północny,
- strzałka w dół (↓) – zmiana kierunku ruchu na południowy,
- strzałka w lewo (←) – zmiana kierunku ruchu na zachodni,
- strzałka w prawo (→) – zmiana kierunku ruchu na wschodni,
- przycisk W – zmiana kierunku ruchu na północny,
- przycisk S – zmiana kierunku ruchu na południowy,
- przycisk A – zmiana kierunku ruchu na zachodni,
- przycisk D – zmiana kierunku ruchu na wschodni.

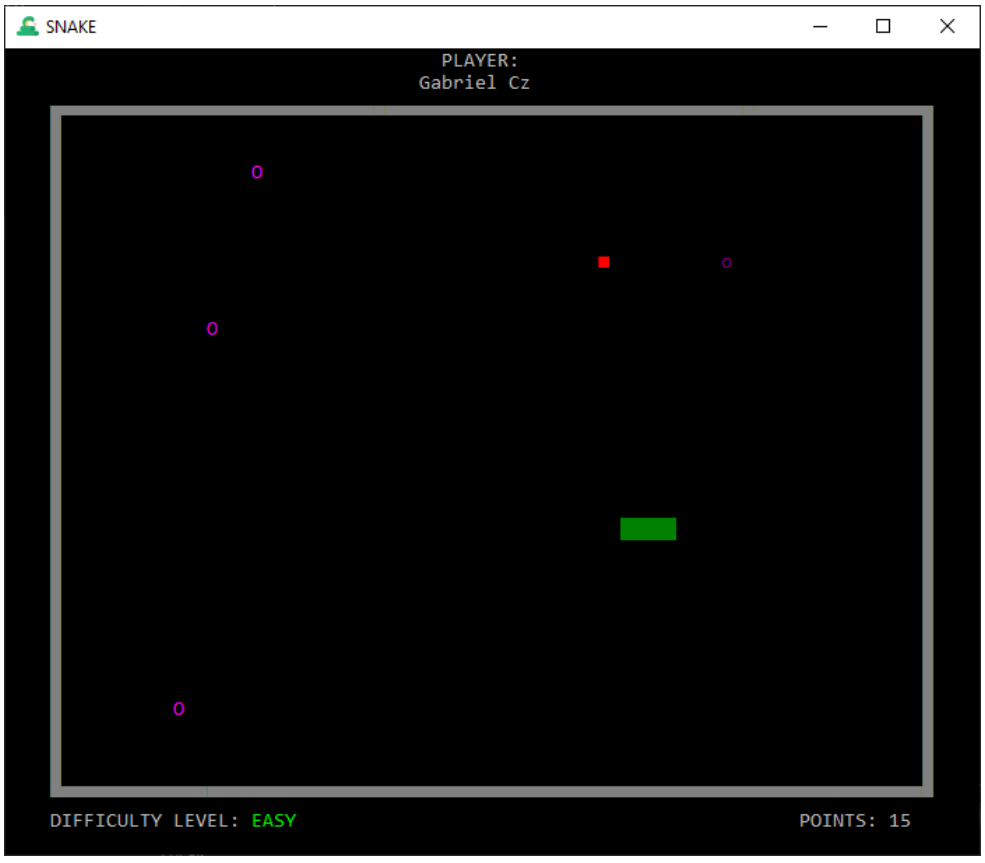
Domyślnym poziomem trudności jej EASY, jednak można go zmieniać w ustawieniach na MEDIUM/HARD lub na własne ustawienia:

Obraz 8. Widok wybierania poziomu trudności.



Obraz 9. Możliwości personalizacji rozgrywki.

W lewym dolnym rogu rozgrywki znajduje się poziom trudności, w prawym dolnym obecna ilość zebranych punktów, a na samej górze nazwa gracza. Standardowe owoce są kwadratami koloru czerwonego (wiśnie), a specjalne (śliwki) to podskakujące literki “o” – ciemny fiolet oraz “O” - jasny fiolet:



Obraz 10. Widok rozgrywki.

Gra kończy się, gdy wąż uderzy głową w ścianę lub część własnego ciała.

W celu sprawdzenia najlepszych wyników należy wybrać z menu opcje “BEST RESULTS”.

Im większy poziom trudności rozgrywki, tym szybszy staje się wąż oraz większa jest ilość punktów zdobywanych za podstawowe pożywienie (EASY – 5, MEDIUM – 10, HARD – 20).

6. WNIOSKI

Projekt spełnia wszystkie wymagania postawione przed jego stworzeniem. Zawiera przyjemny, prosty w obsłudze oraz przejrzysty interfejs tekstowy. Dzięki wykorzystaniu ASCII art imitujących animacje, muzyki grającej w tle oraz opcji personalizacji ustawień aplikacja jest atrakcyjna dla użytkownika końcowego. Odświeżane są tylko niezbędne elementy a nie całe okno co znacznie wpływa na szybkość działania programu. Jest on dynamiczny, nie zacina się i nie ma żadnych błędów.

7. SAMOOCENA

Myślę, że napisana przeze mnie aplikacja działająca w trybie tekstowym spełnia wszystkie postawione wymagania. Jest podobna do oryginału dzięki poświęceniu dużej ilości czasu i wysiłku na jej napisanie i dopracowanie. Nie posiada żadnych błędów oraz działa bardzo szybko. Myślę, że przyciągnęłaby zwolenników gier 8-bitowych.