1.08.2017

Alicja Strąk

[alicjastrak94@gmail.com](mailto:alicjastrak94@gmail.com)

Projekt zaliczeniowy Akademia C#

<https://youtu.be/e7DvAbNHyNk>

1. **Temat projektu.**

Projekt na zaliczenie kursu Akademia c# będzie grą „Snake”, która będzie zrealizowana za pomocą WPF w środowisku Visual Studio.

1. **Funkcjonalności projektu.**

W projekcie zostały wprowadzone wszystkie funkcjonalności opisane w pliku „wstępny opis”.

W menu gry została umieszczona opcja wyboru koloru węża oraz wpisanie nazwy użytkownika. Nie wypełnienie obu pól skutkuje brakiem możliwości rozpoczęcia gry. Po skończonej grze wynik gracza sprawdzany jest rekordowym wynikiem. Jeśli jest większy, zostanie on zapisany i będzie wyświetlany w grze.

W grze „snake” wprowadzono 3 rodzaje pokarmu:

* normal food – jedzenie podstawowe, daje +1 pkt do wyniku oraz wydłużenie węża o jeden segment. Pokarm wyświetlany randomowo w ilości 1-2 szt;
* premium food – daje +10 pkt do wyniku oraz wydłużenie węża o jeden segment. Wyświetlana jest tylko 1 sztuka tego pokarmu, pojawia się co jakiś czas (randomowo);
* dead food – zjedzenie tego pokarmu skutkuje śmiercią węża -> game over.

Obrazy w grze zostały pobrane z gogle graphics.

1. **Inicjalizacja gry.**

W menu gry (MainWindow.xaml) wpisujemy nazwę naszego usera w textBox, Następnie wybieramy kolor węża spośród podanej listy w ComboBox. W przypadku niepoprawnego wprowadzenia danych otrzymamy odpowiedni komunikat MessageBox, by wybrać kolor, jeśli nie został wybrany, wpisać nazwę usera jeśli nie została wpisana lub uzupełnić obie luki. Czcionkę z nazwą gry zaimportowano do projektu (pobrana z internetu).

Po wprowadzeniu poprawnych danych i kliknięciu przycisku start game wyświetla się nowy interfejs (SakeGame.xaml). Po prawej stronie widzimy nasz nick oraz wynik gry, poniżej ustanowiony rekord. Do użycia mamy 3 przyciski, które umożliwiają rozpoczęcie nowej gry, powrót do menu oraz wyłączenie gry.

Gra posiada Interfejs IParticle definiujący wygląd podstawowego elementu występującego w grze. Dziedziczy go klasa Particle. Po klasie Particle dziedziczą klasy NormalFood, PremiumFood, DeadFood oraz PartOfSnake.

Klasy zawarte w grze:

* Klasa Particle: Najbardziej bazowa to klasa Particle, która zawiera lokalizację punktu x i y oraz kształt obiektu – kwadrat o boku 10x10 pikseli. Klasa zawiera metodę Draw(), która umożliwia narysowanie obiektu w siatce gridGame oraz konstruktor.
* Klasa NormalFood: Dziedziczy po klasie Particle, dodatkowo posiada określenie koloru kwadratu – czerwony.
* Klasa PremiumFood: Dziedziczy po klasie Particle, dodatkowo posiada określenie wyglądu kwadratu – wypełniony jest obrazkiem przedstawiającym gwiazdkę.
* Klasa DeadFood: Dziedziczy po klasie Particle, dodatkowo posiada określenie wyglądu kwadratu – wypełniony jest obrazkiem przedstawiającym czaszkę.
* Klasa PartOfSnake: Dziedziczy po klasie Particle i reprezentuje ogon węża. Posiada opcję ustawienia koloru kwadratu na taki, jaki wybrał użytkownik.
* Klasa Snake: zawiera obiekty klasy PartOfSnake. Obiekt headSnake reprezentuje głowę węża a lista partsSnakeList obiekty składające się na ogon węża. W konstruktorze alokujemy pamięć dla obiektu headSnake oraz ustawiamy mu początkowe położenie na planszy gry, standardowy kolor Beige oraz rozmiar kwadratu 12x12 pikseli. Klasa posiada metody:
* DrawSnake(), która rysuje na tablicy całego węża.
* AddParts(), która dodaje nowe części ogona do listy.
* Updatelist(), która zajmuje się przesuwaniem każdego elementu ogona w kierunku przesuwania się głowy(ruch węża)

1. **Działanie gry:**

W konstruktorze okna SnakeGame wywołujemy metody:

* InitializeComponent();
* CreateBoard() – definiuje rozmiar planszy naszej gry (wysokość i szerokość);
* CreateSnake() – tworzy obiekt klasy Snake, dodaje głowę węża i jego ogon do gridGame. Następnie wywołana jest metoda DrawSnake(), która umożliwia wyświetlenie go na planszy gry;
* StartTime() – start czasu gry. Pozwala w interwale czasowym wywoływać metodę TimeUpdate(), która wywołuje metodę UpdateGame();
* MakeFood() – ustawia startowe pozycje pożywienia oraz wyświetla pożywienie na planszy.

Poza tym ustawiamy odpowiednie dane w miejscach na wyniki I nazwy użytkowników.  
W czasie gry co 90 ms wywoływana jest metoda UpdateGame(), która uaktualnia położenie węża, sprawdza kolizję za pomocą metod:

* BoardCollision() – sprawdza kolizję głowy węża z granicami planszy,
* BadColiision() – sprawdza kolizję głowy węża z DeadFood oraz zderzenie ogona z głową, jeśli zostanie wykryte zderzenie gra się zakończy,
* GoodCollision() – sprawdza kolizję z NormalFood oraz PremiumFood, w przypadku kolizji dodawane są punkty do wyniku, wydłużany ogon, losuje nowe położenia dla jedzenia premium, normal oraz dead w przypadku kolizji z NormalFood.
* mySnake.DrawSnake() – rysuje uaktualnione położenie węża.