C# .NET Prosta gra komputerowa - Jeż zbierający owoce

1. Wstęp

Temat zadania 6 to temat dowolny. Postanowiliśmy napisać grę komputerową, która będzie polegała na zbieraniu przez gracza (jeża) jak najwięcej punktów za pomocą zbierania spadających obiektów (owoców). Szczegółowe informacje na temat gry zostaną umieszczone w założeniach. Dodatkowo, interfejs powinien w sposób interaktywny wykorzystywać do komunikacji z operatorem monitor ekranowy oraz klawiaturę.

Do realizacji posłużyliśmy się programem Microsoft *Visual Studio*, *Windows Forms App (.NET Framework)* w języku *C#*, co umożliwia na wykorzystanie techniki graficznego tworzenia interfejsu użytkownika.

2. Założenia

Założyliśmy, że wykorzystanie gry komputerowej będzie przebiegało w następujący sposób:

1) Gracz może poruszać się w prawą lub w lewą stronę, żeby zbierać w koszyk spadające owoce.
2) Zbieranie owoców następuje, gdy obiekt “gracz” zetknie się z obiektem “owoc”.
3) W przypadku, gdy owoc spadnie na ziemię, zamienia się on na plamę.
4) Każdy zebrany owoc, to +1 punkt dla gracza, natomiast każdy stracony owoc to -1 punkt.
5) Owoce spadają ze stałą prędkością, w przypadku gdy przekroczono odpowiednią ilość punktów, owoce generują się i spadają coraz szybciej.
6) Prędkość poruszania się gracza jest stała.
7) Gracz ma 3 życia i każdy stracony owoc zabiera mu 0.5 życia.
8) Gra kończy się, gdy gracz straci wszystkie życia.

3. Postęp pracy

3.1. Wykonanie interfejsu graficznego

Na początku zbudowaliśmy prosty interfejs graficzny. Do tego posłużyliśmy się rozszerzeniem pliku *.cs* który pozwala na tworzenie aplikacji w sposób graficzny. Dodaliśmy 9 *pictureBox’ów:*

- 1-5 są obrazkami różnych owoców;
- 6 - to obrazek jeża;
- 7-9 są obrazkami życia gracza.

Ponadto, dodano 4 *labele*, które zawierają aktualne informacje na temat gry:

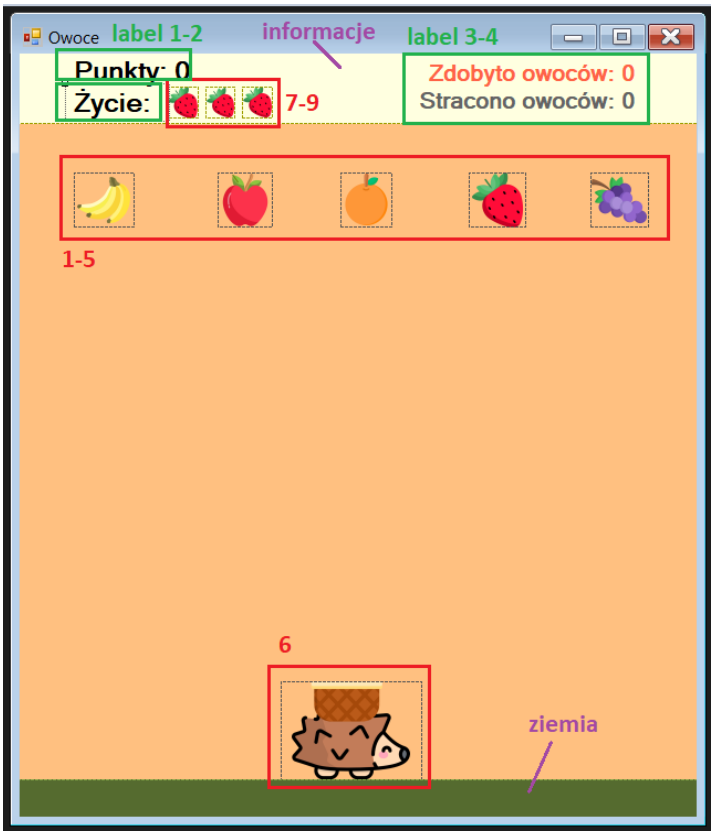
- zdobyte punkty;
- życia gracza;
- ilość zdobytych owoców;
- ilość straconych owoców.

Dodatkowo dodano 2 pola (pusty label), w których jedynie pokolorowano tło, są to:

- pole ziemi;

- pole na zamieszczenie informacji na temat punktacji gry i życia gracza.

Na danym etapie plik *Form1.cs [Design]* wyglądał następująco:



Rys.1 - Zbudowanie interfejsu graficznego

3.2. Podstawowe funkcje kodu

Aby program działał według założeń, stworzyliśmy podstawowe funckje, których zadaniem jest:

1. resetowanie gry;
2. poruszanie się graczem po ziemi;
3. upuszczanie owoców;
4. interakcja gracza z owocami - punktacja i życie.

Wszystkie interakcje obiektów są możliwe jedynie w przypadku, gdy uruchomiono timer (*timer.Start()* w f-cji *Restart()*). Są one zamieszczone w funkcji *TimerGry()*.

Na początku zadeklarowaliśmy zmienne, którymi będziemy posługiwać się w programie. Listę zmiennych wraz z ich zastosowaniem przedstawiono na poniższym rysunku.

```
bool idzLewo, idzPrawo;
int predkosc; // prędkość spadania owoców
int punkty; // zdobyte punkty
int zdobyto; // zdobyto owoców
int stracono; // stracono owoców
int zycie_gracza;
// na sam początek gracz ma 3 życia
//(-1 życie to 2 stracone owoce)


Random randX = new Random(); // położenie X owoców
Random randY = new Random(); // położenie Y owoców

// po straceniu owoca pojawia się plama
PictureBox plama = new PictureBox();
```

Rys.2 - Inicjalizacja zmiennych

Restartowanie gry odbywa się za pomocą funkcji *Restart()*, która jest wywołana przy każdym uruchomieniu gry (pierwszym i ponownym). Dana funkcja zeruje wszystkie zmienne, ustawia obiekty w pozycję początkową oraz uruchamia timer gry.

```
// f-cja restartuje grę
private void Restart()
{
    // wszystkie obiekty ktore są owocami są przesunięte w górę
    foreach (Control x in this.Controls)...

    // gracz znajduje się po środku
    gracz.Left = this.ClientSize.Width / 2;
    gracz.Image = Properties.Resources.jez1;

    // zerujemy zmienne
    punkty = 0;
    zdobyto = 0;
    stracono = 0;
    predkosc = 4;
    zycie_gracza = 6;

    // jeż stoi w miejscu
    idzLewo = false;
    idzPrawo = false;

    // wznawiamy życie gracza (x3 truskawki)
    zycie3.Image = Properties.Resources.truskawka;
    zycie3.Visible = true;
    zycie2.Image = Properties.Resources.truskawka;
    zycie2.Visible = true;
    zycie1.Image = Properties.Resources.truskawka;
    zycie1.Visible = true;
    // usuwamy plamę
    plama.Visible = false;
    // timer gry start
    timer.Start();
}
```

Rys.3 - Funkcja Restart(), restartująca grę

Kolejną niezbędną funkcją jest funkcja poruszania się gracza za pomocą strzałek na klawiaturze. Służą do tego funkcje *NaDole()* i *NaGorze()*, które reagują na naciśnięcie klawiszy strzałek (<-- –>) i ustawiają zmienne *idzLewo* i *idzPrawo* na *true/false*. Mając różne konfiguracje zmiennych *idzLewo, idzPrawo*, gracz albo przesuwa się w lewą stronę, albo w prawą, albo stoi w miejscu. Przesuwanie się gracza zamieszczone jest w części funkcji

TimerGry() (rys.5).

```
// f-cje pozwalające na przemieszczenie jeża za pomocą klawiatury
private void NaDole(object sender, KeyEventArgs e)...
private void NaGorze(object sender, KeyEventArgs e)...
```

Rys.4 - Funkcje NaDole(), NaGorze() reagujące na naciśnięcie strzałek klawiatury

```
// jeśli nacisnięto strzałki, jeż przemieszcza się o 12 pikseli, jeśli nie napotkano się na ścianę
if (idzLewo == true && gracz.Left > 0)...
if (idzPrawo == true && gracz.Left + gracz.Width < this.ClientSize.Width)...
```

Rys.5 - Funkcja TimerGry(), warunki,

Po tym jak w *Restart()* załączono timer, wykonuje się działanie funkcji *TimerGry().* Na samym początku są wypisane informacje na temat aktualnych punktów gracza, jego stanu zdrowotnego oraz liczby zdobytych i straconych punktów. Dane dane są wprowadzane w labele. Informacje te będą odnawiane przy każdym straconym lub zdobytym owocu (rys.5).

```
// jak załącza się timer, wykonują się działania w f-cji TimerGry
private void TimerGry(object sender, EventArgs e)
{
    ile_punktow.Text = "Zdobyto owoców: " + zdobyto;
    ile_stracono.Text = "Stracono owoców: " + stracono;
    zycie.Text = "Życie: ";
    pkt.Text = "Pukty: " + punkty;
```

Rys.5 - Początek funkcji TimerGry()

Najciekawszą częścią danej funkcji jest ciąg dalszy, gdzie przedstawiona jest interakcja gracza z obiektami (owocami) oraz napisane są różne warunki pozwalające na płynną działalność gry. Wgłębijmy się w zawartość *foreach()* (rys. 6).

Na sam początek przedstawiono, co będzie się działo, gdy owoc nie zostanie złapany przez gracza *(if(x.Top + x.Heigh +20 > this.ClientSize.Height)).*

1. Ustalono, że **x** będą oznaczane wszystkie owoce, spadające z prędkością *prędkość*.
2. Jeśli owoc spadnie na ziemię, to na miejscu owoca **x** pojawi się *plama*, a owoc (obiekt) zostanie wygenerowany w losowym miejscu na górze i za kilka sekund ponownie spadnie i będzie widoczny dla gracza.
3. Konsekwencje straconych/ zebranych owoców są takie, że jeśli gracz straci owoc, to jego punktacja zmniejszy się o -1 i jego życie (obrazek truskawki) zmaleje o 0,5 (*switch(zycie_graca)* - obrazek pełnej truskawki zmienia się na obrazek jej połowy).
4. Jeśli stracono owoc, obrazek gracza (stojącego jeża) zamienia się na jeża siedzącego. Aby ponownie zacząć zbierać owoce, trzeba nacisnąć strzałkę “w prawo” lub “w lewo”.

```
foreach (Control x in this.Controls)
{
    if (x is PictureBox && (string)x.Tag == "owoce")
    {
        x.Top += predkosc; // owoce poruszają się z prędkością predkosc

        // jeśli owoce spadły na ziemię, to zamieniają się w plamy
        if(x.Top + x.Height +20 > this.ClientSize.Height)
        {
            plama.Visible = true;
            plama.Image = Properties.Resources.plama2;
            plama.Location = x.Location; // plama pojawia się na miejscu owoca który spadł
            plama.Height = 60; //parametry obrazka plamy
            plama.Width = 60;
            this.plama.SizeMode = PictureBoxSizeMode.Zoom;
            plama.BackColor = Color.Transparent;

             this.Controls.Add(plama); //dodano plamę do wyświetlanej gry

            x.Top = randY.Next(80,300) * (-1); // jeśli owoc spadł to generuje się nowy na górze
            x.Left = randX.Next(5, this.ClientSize.Width - x.Width);
            stracono += 1;// traci się punkty i życie gracza
            punkty -= 1;

            if (punkty <= 0)...

            zycie_gracza -= 1;

            switch (zycie_gracza)...

            gracz.Image = Properties.Resources.jez3;// zmienia się obrazek jeża gdy gracz straci owoc
        }

        if(gracz.Bounds.IntersectsWith(x.Bounds))...
    }
}
```

Rys.6 - Interakcja gracza i owoca - warunek: nie złapano owoca

W przypadku, gdy gracz zbierze owoc w koszyk (obiekt “owoc” dotknie obiekt “gracz”) (*if(gracz.Bounds.IntersectsWidth(x.Bounds)))*, gracz dostaje +1 punkt, a owoc generuje się losowo na górze i za jakiś czas ponownie spadnie i będzie widoczny dla gracza (rys.7).

```
if(gracz.Bounds.IntersectsWith(x.Bounds))
    // jeśli gracz dotknie owoc, to gracz zdobywa owoc
{
    x.Top = randY.Next(80, 300) * (-1); // owoc generuje się na górze
    x.Left = randX.Next(5, this.ClientSize.Width - x.Width);
    zdobyto += 1; // naliczają się pukty za zdobyty owoc
    punkty += 1;
}
```

Rys.7 - Interakcja gracza i owoca - warunek: złapano owoc

Po tym, jak gracz zdobędzie X owoców, zostanie zwiększona prędkość. Prędkość wzrasta odpowiednio do wartości zebranych owoców (rys.8).

```
// po przekroczeniu x zdobytych owoców zwiększa się prędkość spadania owoców
if(zdobyto >= 10)
{
    predkosc = 5;
}
```

Rys.8 - Część kodu przedstawiająca wzrost prędkość gdy zebrano 10 owoców

Gra kończy się w momencie, gdy gracz straci wszystkie swoje życia. Wtedy zostają podsumowane punkty oraz liczba zdobytych i straconych owoców. Ostatecznie wyskakuje *MessageBox* informujący gracza o końcu gry i proponujący zagrać ponownie.

```
// jeśli gracz stracił wszystkie życia, koniec gry
if (zycie_gracza <= 0)
{
    ile_punktow.Text = "Zdobyto owoców: " + zdobyto;
    ile_stracono.Text = "Stracono owoców: " + stracono;
    pkt.Text = "Pukty: " + punkty;
    zycie.Text = "Życie: 0";
    gracz.Image = Properties.Resources.jez3;

    timer.Stop(); // timer stop
    MessageBox.Show("Koniec Gry!" + Environment.NewLine + "Zmęczyłeś jeża :("
        + Environment.NewLine + "Naciśnij OK żeby spróbować ponownie.");
    Restart();
}
```

Rys.9 - Interakcja gracza i owoca - warunek: koniec gry (stracono 6 owoców)

4. Wyniki pracy

Wyniki pracy można uznać za pomyślne, gdyż spełniono wszystkie założenia zadania. Dana aplikacja dobrze pełni swoją funkcję gry, nie występują błędy lub bugi. Interfejs graficzny gry oraz sam kod jest prosty, czytelny i uwzględniający wszystkie funkcje przyjęte w założeniach.

Można uznać, że zaletą tej gry jest jej prostota budowy oraz łatwość obsługi, natomiast wadą jest to, że dana gra nie jest rozbudowana. Nie uwzględnia ona dodatkowych funkcji, na przykład bonusów, zmiany tła, podziału gry na etapy lub poziomy, i może szybko znudzić się graczowi. Jednak z założenia, dany program nie miał być skomplikowany, a dodawanie dodatkowych funkcji jest kwestią fantazji i wymyślności.

Najwięcej trudności doświadczyliśmy podczas definiowania interakcji gracza z owocami (obiektami typy pictureBox). Trzeba było zrozumieć zasadę działania timera oraz odpowiednio dobrać warunki, w których owoce są zbierane/tracone. Również nie było łatwe wymyślenie sposobu, w jaki program generował i przesuwał owoce. Jednak po przeprowadzeniu analizy oraz poszerzeniu wiedzy na dany temat udało nam się stworzyć w pełni działający i funkcjonalny program.