Projekt zespołowy

Prowadzący: Janusz Rafałko

Członkowie: Rafał Dobrowolski, Wiesław Bikowski, Kacper Buczkowski

Temat projektu

Gra typu Space-shooter, utrzymana w klimacie retro

Spis treści:

1. Cele biznesowe/cel projektu.

Podstawą projektu jest stworzenie gry typu space-shooter utrzymanej w retro-klimacie, lecz przy użyciu nowych technologii i sposobu projektowania aplikacji. Produkt tworzony jest ze względu na wzrost zainteresowania powrotem do gier typu retro oraz popularyzacji grafiki pixelowej. Jest to projekt przeznaczony dla użytkowników w każdej ramie wiekowej, gdyż młodzi uznać mogą go za interesujący a starsi za nieskomplikowany i przystępny.

Aplikacja skierowana jest na systemy typu Windows oraz Android. Dodatkowo stworzona zostanie strona internetowa przedstawiająca podstawowe informacje o grze (promocja gry) oraz umożliwiająca pobranie odpowiedniej wersji(desktop/mobile).

2. Cel projektu

Dzięki temu przedsięwzięciu rozwiniemy swoje zdolności w dziedzinie programowania oraz działania w zespole jak i umiejętność odpowiedniego zarządzaniem czasem. Stworzenie nawet pozornie prostej gry wymagać będzie od nas zaznajomienia się z podstawą funkcjonowania i konstruowania mechanik, tworzeniem grafiki oraz dźwięku i ostatecznie składania wszystkiego w jedną w pełni funkcjonującą całość.

3. Funkcje projektu

Akcja	Czynność użytkownika	Reakcja systemu
Gra		
Uruchomienie gry	Wciśnięcie ikony gry	Uruchomienie gry, ekranu początkowego i przejście do menu. Wyświetlenie wcześniej zapisanego lokalnie najlepszego wyniku.
Sterowanie	Wciśnięcie strzałek(desktop) bądź przyłożenie palca do ekranu(mobile).	Przesuwanie statku gracza w obranym kierunku.
Nawigacja	Wciśnięcie wybranego przycisku.	Zależnie od wybranej opcji: wyjście, start, pauza, wznowienie gry.
Wyświetlenie wyniku	Zakończenie gry.	Wyświetlenie najlepszego zdobytego wyniku.
Koniec gry	Utrata trzech żyć.	Przeniesienie gracza do menu głównego.
Zamknięcie gry	Wciśnięcie przycisku exit	Zapis lokalny zdobytego wyniku
Strona internetowa		
Prezentacja gry	Otworzenie strony internetowej.	Wyświetlenie informacji o grze i autorach oraz prezentacja rozgrywki.
Ściągnięcie gry	Kliknięcie przycisku do pobrania.	Pobranie na urządzenie użytkownika odpowiedniej - wcześniej wybranej wersji gry.
Nawigacja	Wybór opcji nawigacji.	Przeniesienie do odpowiedniej sekcji strony.

Tabela 1: tabela przedstawiająca podstawowe funkcje projektu

4. Wymagania

a) Wymagania funkcjonalne

Gra	
Stworzenie gry na desktop i mobile	Gra będzie space-shooterem w pixelowej retro grafice.
Ekran dotykowy	Statek na mobile przesuwany palcem
Kontrola sterowania	Statek nie może wychodzić poza granice planszy i porusza się jedyni na osi poziomej.
Zapis wyniku	Wynik zapisywany jest lokalnie i wczytywany po ponownym załadowaniu gry
Funkcjonalne menu	Przygotowanie menu gry zanim przejdzie się bezpośrednio do rozgrywki. Start, wyjście pauza itp. Być może rozbudowa o dodatkowe opcje.
Strona internetowa	
Stworzenie strony poświęconej grze	Strona stanowić będzie reklamę gry oraz pozwoli na jej ściągnięcie w wybranej wersji.
Responsywność	Strona dopasowywać będzie się do różnych rozdzielczości ekranów - komputerów i telefonów.
Prezentacja	Na stronie prezentowane będą grafiki lub filmy z przykładowej rozgrywki.
Spójność z grą	Strona utrzymana będzie w podobnej szacie graficznej co gra.

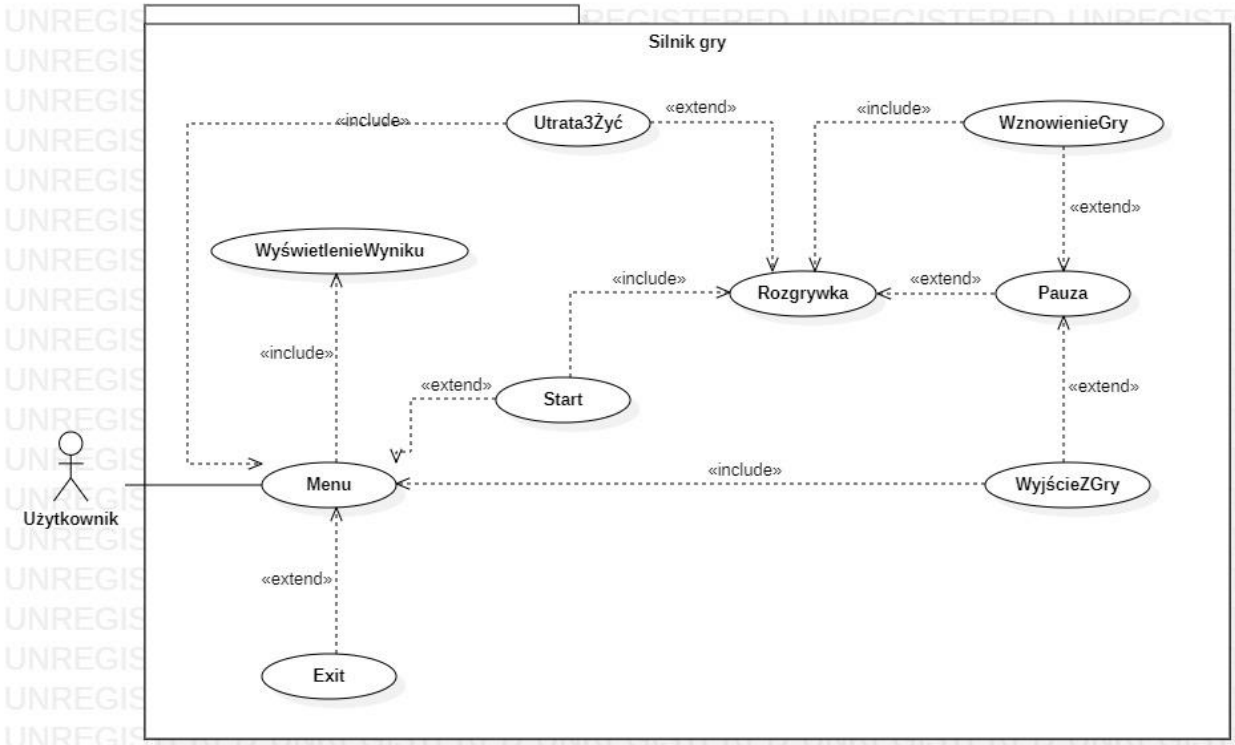
Tabela 2: tabela przedstawiająca wymagania funkcjonalne projektu

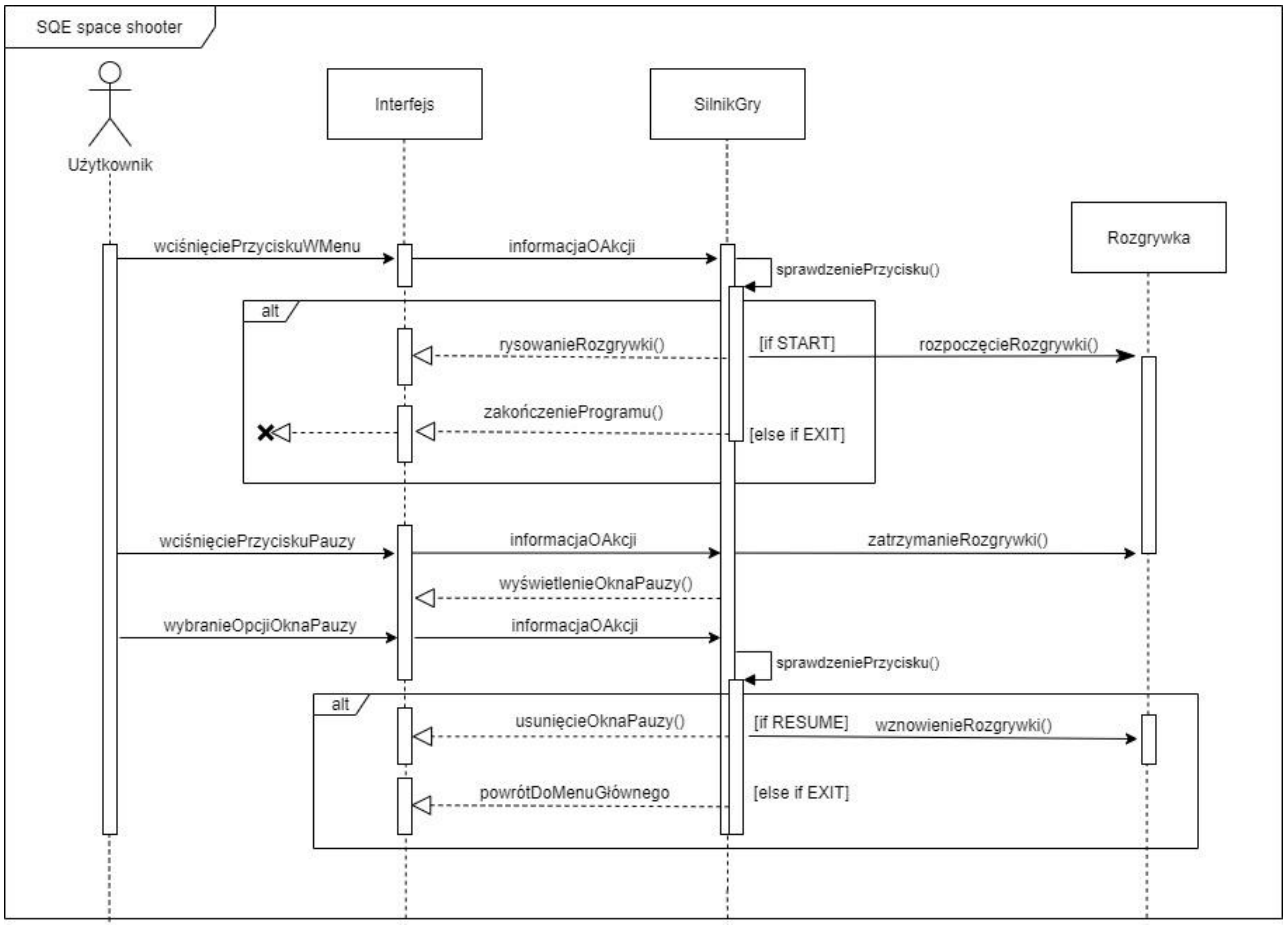
b) Wymagania niefunkcjonalne

Gra	
Wymagania sprzętowe	Komputer z systemem Windows lub telefon komórkowy z systemem android, klawiatura, monitor.
Niezawodność	Z racji prostoty tworzonej gry nieprzewidywane są żadne utrudnienia związane z niezawodnością.
Łatwość użytkowania	Aplikacja jest bardzo łatwa w użytkowaniu dla użytkowników w każdym przedziale wiekowym.
Wydajność	Aplikacja jest dobrze zoptymalizowana, dzięki czemu może zostać uruchomiona nawet na słabym sprzęcie.
Przenośność	Docelowo gra projektowana jest na platformy Windows oraz Android, aczkolwiek poprzez zastosowanie frameworku Monogame możliwe jest przenoszenie produktu na dowolne urządzenie.
Strona internetowa	
Wymagania sprzętowe	Komputer lub telefon z zainstalowaną aktualną przeglądarką internetową.
Niezawodność	Uzależniona od stabilności serwera, na którym strona jest postawiona.
Łatwość użytkowania	Użytkownik nie będzie czuł się zagubiony czy też przytłoczony nadmiarem niepotrzebnych informacji. Prosto, schludnie i na temat o tworzonej aplikacji.
Wydajność	Strona uruchamia się szybko oraz jest zoptymalizowana pod względem działania i prędkości reakcji.
Przenośność	Strona kompatybilna z większością popularnych przeglądarek internetowych.

Tabela 3: tabela przedstawiająca wymagania niefunkcjonalne projektu

c) Diagram przypadków użycia




SQE space shooter
Użytkownik
Interfejs
SilnikGry
Rozgrywka
wciśnięciePrzyciskuWMenu
informacjaOAkcji
sprawdzeniePrzycisku()
alt
rysowanieRozgrywki()
[if START]
rozpoczęcieRozgrywki()
zakończenieProgramu()
[else if EXIT]
wciśnięciePrzyciskuPauzy
informacjaOAkcji
zatrzymanieRozgrywki()
wyświetlenieOknaPauzy()
wybranieOpcjiOknaPauzy
informacjaOAkcji
sprawdzeniePrzycisku()
alt
usunięcieOknaPauzy()
[if RESUME]
wznowienieRozgrywki()
powrótDoMenuGłównego
[else if EXIT]

5. Technologia

W celu wykonania założonej gry wykorzystujemy język C# wraz z frameworkiem Monogame oraz środowiska programistycznego Visual Studia. Strona internetowa wykonywana jest w środowisku Visual Studio Code w oparciu o HTML, CSS, JS, Jquery oraz hostowana będzie na domenie cba.pl. Wykorzystywane grafiki tworzone są w programie Aseprite.

6. Metodyka

W pisaniu aplikacji najbardziej odpowiada nam metodyka przyrostowa, gdyż mamy określony zarys aplikacji a kolejne funkcjonalności dodawać będziemy w trakcie powstawania projektu. Pozwala to nam również na wdrażanie nowych pomysłów, których wcześniej nawet nie rozważaliśmy. Wykonując poszczególne fragmenty gry możemy je testować by następnie w razie konieczności dokonać ich poprawy bądź przejść do tworzenia kolejnych funkcjonalności.

7. Podział pracy

Ze względu na nasze umiejętności podział pracy wygląda następująco:

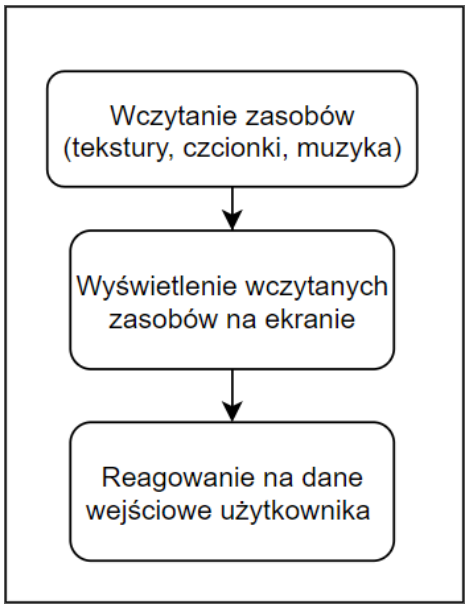
- Rafał Dobrowolski – lider grupy, strona internetowa, hosting itp.,
- Wiesław Bikowski – aplikacja dektopowa i grafika,
- Kacper Buczkowki – główny programista, aplikacja mobila.

8. Harmonogram

06.04.2020	Przygotowanie większości grafik wymaganych do pozostałych części projektu. Konfiguracja środowisk programistycznych oraz wstępne stworzenie projektu.
20.04.2020	Stworzenie wersji alpha strony internetowej oraz projektowanej gry. Zaprogramowanie funkcjonalnego menu oraz podstaw rozgrywki.
27.04.2020	Rozbudowa rozgrywki i strony internetowej o dodatkowe elementy.
04.05.2020	Rozbudowa rozgrywki i strony internetowej o dodatkowe elementy.
11.05.2020	Deadline. Stworzenie większości zakładanych elementów. Gra – zliczanie punktów, zapis lokalny wyniku, zróżnicowanie przeciwników oraz statków do wyboru itp..
18.05.2020	Rozbudowa rozgrywki i strony internetowej o dodatkowe elementy.
25.05.2020	Deadline. Wersja grywalna gry oraz w pełni funkcjonująca strona internetowa
01.06.2020	Dopracowanie projektu
08.06.2020	Dokumentacja techniczna projektu
15.06.2020	Gotowy projekt

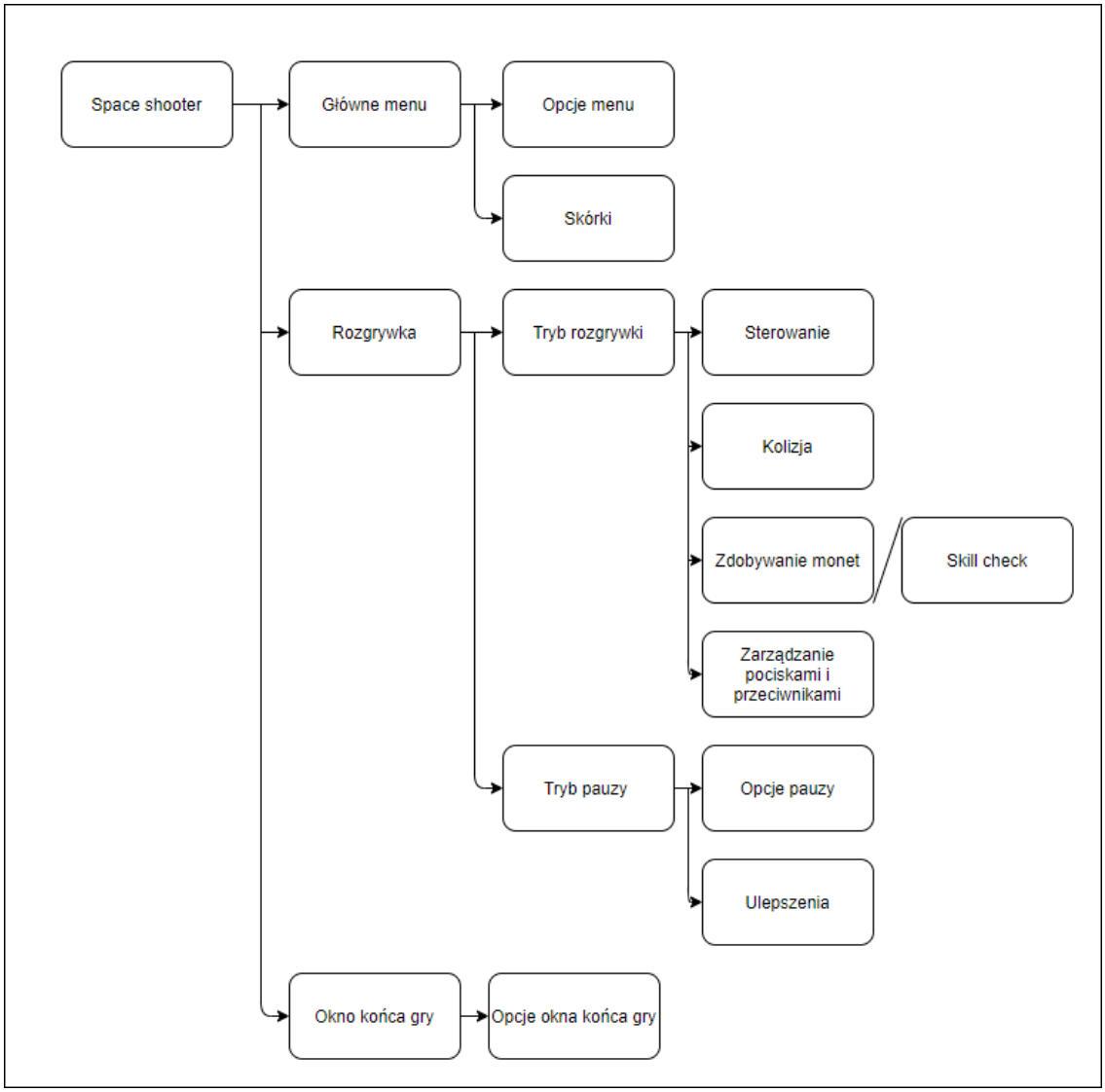
Tabela 4: Harmonogram wykonywanego projektu

9. Model systemu



Rysunek 3: ogólny model systemu

Rysunek 3 przedstawia możliwie najbardziej ogólny model tworzonego systemu. Założenie tworzenia gry jest proste - wczytać potrzebne zasoby, użyć je zależnie od typu ich typu, następnie w razie otrzymania danych wejściowych gracza (kliknięcie myszy czy wciśnięcie strzałki bądź innego przycisku klawiatury) zmienić właściwości danego zasobu (np. położenie). Framework Monogame nadaje się idealnie do zaprezentowanego systemu (więcej informacji w części implementacji).



Rysunek 4: szczegółowy model sytemu

Rysunek 4 przedstawia konkretne elementy wymagające implementacji.

Rozróżnić tu można trzy główne części: menu główne, rozgrywkę oraz okno końca gry. Każda z nich posiada pewne opcje do wyboru z poziomu dedykowanego dla danego elementu menu. Tak dla przykładu główne menu składa się z opcji menu – START, EXIT oraz SKINS gdzie ten ostatni jest odrębnym elementem o innej strukturze i działaniu niż przycisk startu i wyjścia z programu. Najbardziej obszernym modułem okazał się być moduł wykrywania i zarządzania wszelkiego rodzaju kolizją, gdyż to on odpowiadał za min.: kolizję pocisków i wrogów, zliczanie punktów, odejmowanie punktów życia itd.

10. Implementacja

```
enum GameState
{
    MainMenu,
    GamePlay,
    EndOfGame,
}
GameState _currentGameState;
```

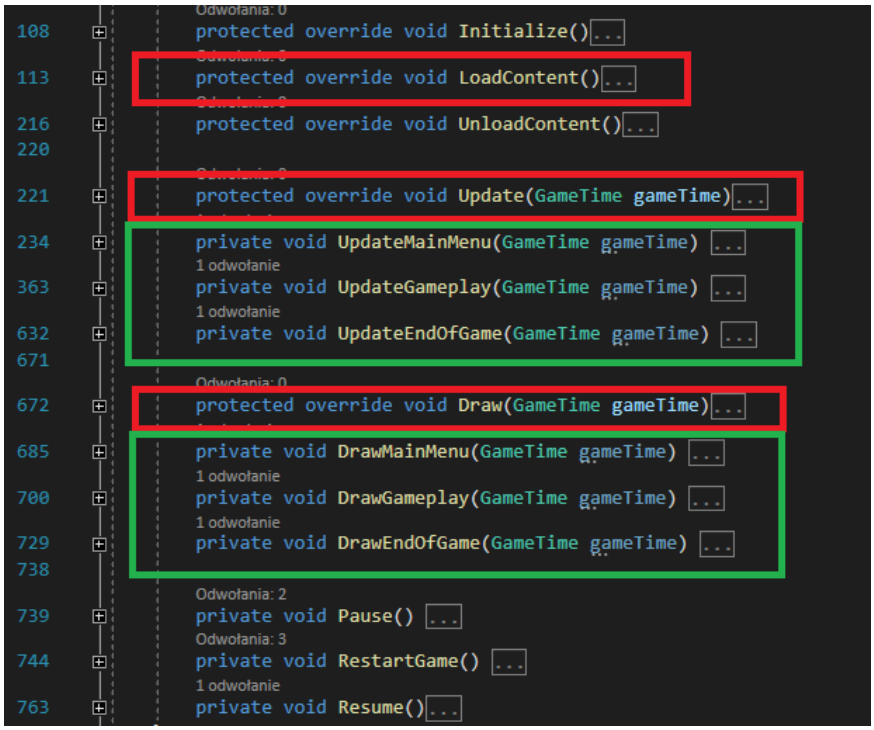
Rysunek 5: Typ wyliczeniowy sterujący trybem gry

Zgodnie z rysunkiem 4, gra opiera się na działaniu w trzech różnych stanach - główne menu, rozgrywka oraz okno końca gry. Wykorzystanie typu wyliczeniowego pozwoliło w łatwy sposób się po między nimi przemieszczać co znacznie ułatwiło nawigację programu i zarządzanie kodem. Każdy z poszczególnych trybów odpowiada za inną część gry.

MainMenu zawiera okno początkowe oraz menu pozwalające rozpocząć rozgrywkę, zamknąć program czy też przejść do panelu z wyborem skórki statku.

GamePlay steruje rdzeniem rozgrywki - wyświetlaniem gracza, przeciwników, pocisków, zarządzania kolizję oraz obsługą sterowania, zlicza i wyświetla punkty zdobyte za zestrzeliwanie wrogów, odejmuje punkty życia gracza oraz przełącza stopień trudności rozgrywki zależnie od ilości posiadanych punktów. Część ta steruje również oknem pauzy, gdzie mamy opcje takie jak wznowienie, restart, powrót do ekranu startowego oraz w przypadku wersji desktopowej – okno ulepszeń pozwalające rozwinąć statek gracza poprzez wydawanie zdobytej podczas rozgrywki waluty.

EndOfGame służy jako część wyświetlająca się, gdy gracz zakończy rozgrywkę. Pokazuje ilość zdobytych punktów oraz umożliwia szybkie zaczęcie rozgrywki od zera lub powrót do głównego menu.



Rysunek 6: Podstawowe funkcje

Rysunek 6 przedstawia sposób implementacji założenia przedstawianego na rysunku 3 - ogólne model systemu. Zgodnie z nim mamy trzy podstawowe funkcje (kolor czerwony) odpowiadające za:

- ładowanie zasobów - LoadContent(),
- wyświetlanie zasobów - Draw(),
- reagowanie na dane wejściowe - Update().

Funkcje Draw oraz Update wywoływane są z pewną częstotliwością co pozwala na ciągły monitoring zdarzeń czy też płynne wyświetlanie zasobów.

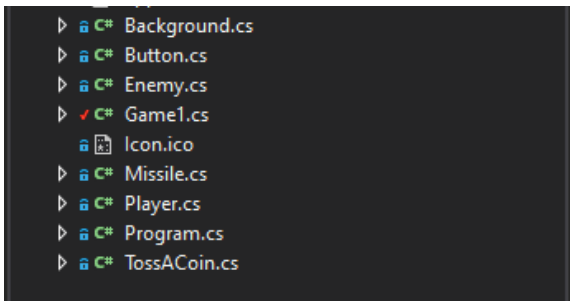
```
protected override void Update(GameTime gameTime)
{
    //if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonSt

    switch (_currentGameState)
    {
        case GameState.MainMenu: UpdateMainMenu(gameTime); break;
        case GameState.GamePlay: UpdateGameplay(gameTime); break;
        case GameState.EndOfGame: UpdateEndOfGame(gameTime); break;
    }

    base.Update(gameTime);
}
```

Rysunek 7: Struktura funkcji update

W celu ułatwienia poruszanie się w kodzie funkcja Update odpowiada jedynie za przełączenie do odpowiedniej funkcji zarządzającej tylko i wyłączenie jednym modułem programu. Dla przykładu - jeżeli obecny stan gry to rozgrywka (GameState.GamePlay), funkcja Update przełącza sterowanie do podrzędnej funkcji (na rysunku 6 – kolor zielony) UpdateGameplay. Analogicznie dzieje się w przypadku funkcji Draw.

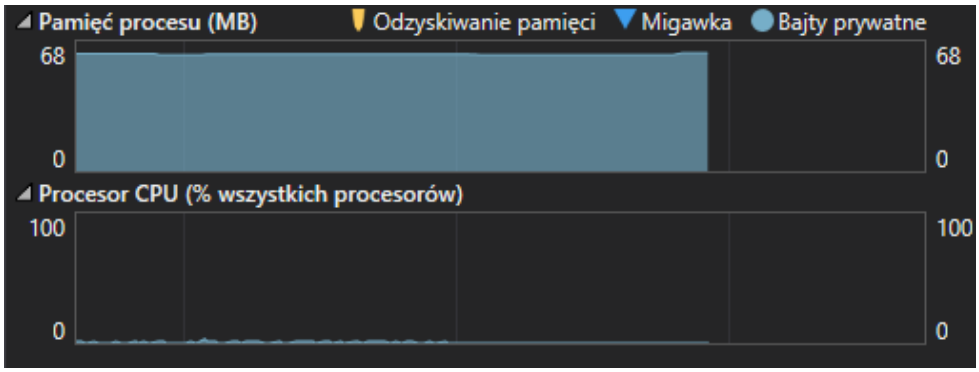


Rysunek 8: Klasy programu

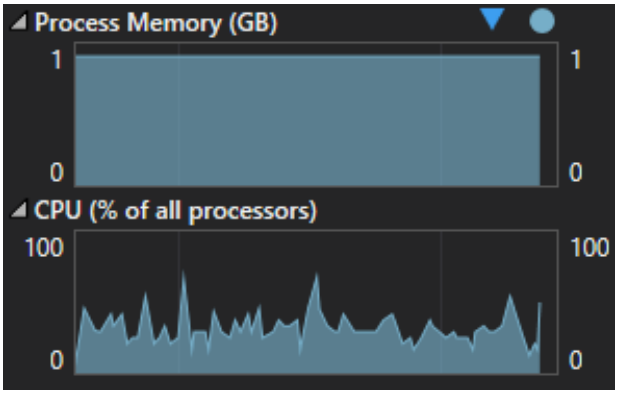
Gra składa się z kilki interaktywnych i odrębnych obiektów takich jak przyciski, statek gracza, tło, pociski, wrogowie czy monety. Każdy z nich ma dedykowaną klasę składającą się zazwyczaj z pola Texture przechowującego teksturę obiekty oraz Rectangle określającego położenie obiektu. Dodatkowo każda klasa posiada metody wymagane do funkcjonowania obiektu. Tak dla przykładu klasa Player zawiera metody UpdateMovement(), DrawHearths() oraz Hit()

11. Testy

Gra odporna jest większość błędów ze względu na bardzo ograniczony wachlarz danych wejściowy i możliwości interakcji. Co do samego uruchomienia – wersja desktopowa testowana była na kilku urządzeniach z systemem Windows 10 z aktualnym .NET Framework-iem i nie zauważono żadnych nieprawidłowości czy trudności z uruchomieniem. Te ostatnie pojawiły się podczas próby uruchomienia gry na Windows 7.



Rysunek 8: Zużycie zasobów - wersja desktop



Rysunek 9: Zużycie zasobów - wersja mobile

Gra została również zoptymalizowana pod względem zużycia zasobów i zabezpieczona przed niekontrolowanym zużyciem pamięci. Bardzo duże zużycie zasobów na urządzeniu mobilnym(Rysunek 9) spowodowany jest koniecznością emulowania tegoż to urządzenia.

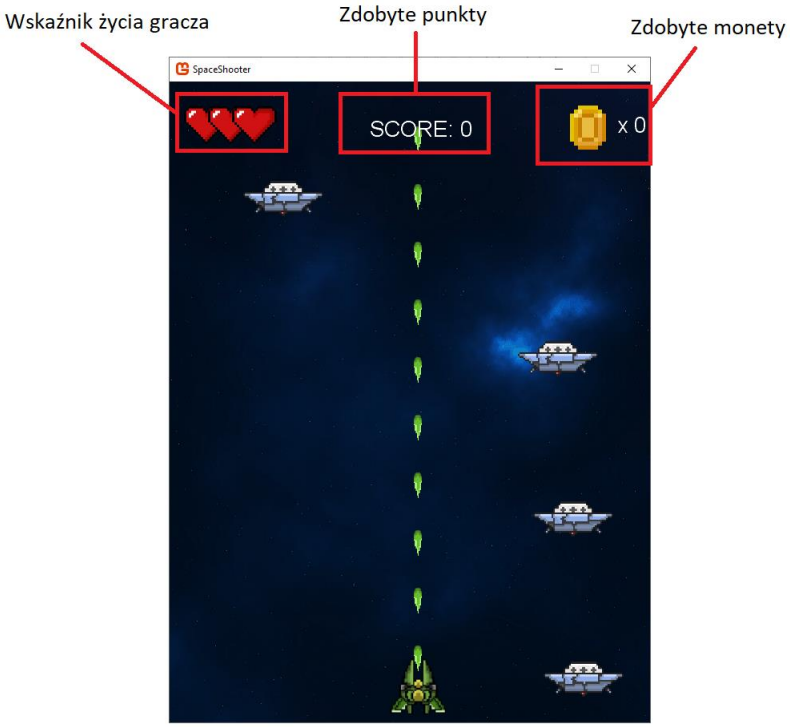
Optymalizacji poddane zostały np. pociski oraz przeciwnicy. Obie klasy mają pole isDestroyed które taguje obiekty które można już usunąć z pamięci (np. zostały zestrzelone lub wyleciały za obszar ekranu).

12. Instrukcja użytkowania

Rysunek 10: Główne menu

Rysunek 11: Okno skórek

Po włączeniu gry ukaże się ekran startowy wraz z trzema opcjami do wyboru. START umożliwia nam przejście bezpośrednio do rozgrywki, SKIN umożliwia wybór alternatywnego wyglądu statku, EXIT pozwala zamknąć aplikację. Z okna skórek można wyjść naciskając przycisk “ESC”.

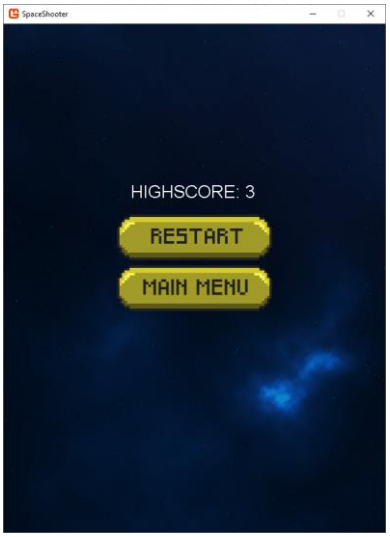


Rysunek 12: Okno rozgrywki

Rysunek 13: Okno pauzy

Rysunek 14: Okno ulepszeń

Okno pauzy uruchamiamy wciskając podczas rozgrywki przycisk “ESC”. Z opcji możemy kolejno wznowić rozgrywkę, zacząć od nowa, przejść do okna ulepszeń lub wyjść do menu głównego. Uruchamiając okno ulepszeń możemy wykupić rozbudowę statku wydając zebrane wcześniej monety. Ulepszenie kupujemy klikając na nie.



Rysunek 15: Okno końca gry

Okno końca gry wyświetla się po utracie trzech żyć. Umożliwia ono szybkie wznowienie gry lub przejście do głównego menu. Dodatkowo pokazuje ono liczbę zdobytych punktów.

13. Podsumowanie

- Cele zrealizowane / niezrealizowane

  Stanowczą większość zakładanych celów udało się zrealizować bez większych problemów a także udało się zaimplementować wiele mechanizmów które początkowo nie były brane pod uwagę (system ulepszeń oraz skill check). Udało się stworzyć w pełni grywalną i funkcjonalną wersje desktop dla systemów Windows 10 oraz mobile dla telefonów z systemem android. Jedynym mankamentem, który wymagałby dopracowania to sama rozgrywka. Miała ona mocno opierać się na zręczności gracza, lecz finalny produkt nie do końca spełnia to założenie. Gra wymagałaby gruntownej przebudowy by to zmienić. Nie udało nam się również popracować z animacjami (np. wybuchy, animowane pociski itp..) gdyż silnik, na którym pracowaliśmy miał co to tego spore ograniczenia. Przesiadka np. Na platformę UNITY znacznie by to ułatwiła.
- Możliwe kierunki rozwoju

  Przesiadka na platformę UNITY była by dobrym pomysłem. W łatwy sposób możny by było rozwinąć animacje oraz ulepszyć sprawdzanie kolizji (Monogame oferuje to w ograniczonym zakresie). Co do samej gry warto by było dodać różne trajektorię lotu przeciwników, większą ich różnorodność oraz o czym wspomniałem wcześniej, wynajmować ich ruch i kolizję. Znacznie by to ulepszyło wizualny odbiór rozgrywki. Można by było się zastanowić nad modelem sieciowym – nad jakąś prostą rywalizacją w zdobywaniu punktów i ewentualnych nagrodach.
- Wnioski

  Korzystając z frameworku Monogame nauczyliśmy się naprawdę sporo na temat tworzenia gier i ich mechanizmów. Tworząc projekt skupialiśmy się na obiektowym podejściu do rozwiązywania problemów co w znaczącym stopniu rozwinęło nasze umiejętności programistyczne.

  Praca grupowa przebiegała bez przeszkód z racji jasnego i konkretnego podziału zadań do wykonania. Zauważyliśmy również pisząc dwie wersje - desktop i mobile - zdrową napędzająca pracę konkrecje.