Politechnika Białostocka

Sprawozdanie z pracowni specjalistycznej

Zaawansowane techniki programistyczne

Raport z projektu

Temat: **Statki kosmiczne**

Wykonujący ćwiczenie:

Michał Ziółkowski

Dawid Pawłowski

Studia dzienne

Kierunek: Informatyka i ekonometria

Semestr: V

Grupa zajęciowa:PS 2

Prowadzący ćwiczenie: **mgr inż. Daniel Reska**

Data wykonania ćwiczenia:

28.01.2021

Końcowy opis projektu

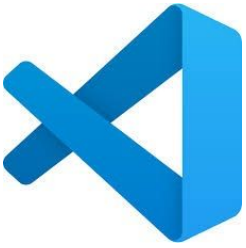
Space shooter: to gra z interfejsem graficznym w której wcielamy się w rolę pilota który poluje na kosmiczne statki w kosmosie. Naszym zadaniem jest pokonanie jak największej liczby przeciwników. Walka odbywa się w sposób ciągłły.W trakcie trwania rozgrywki gracz może poruszać się po osi X(lewo, prawo). Za pokonywanie przeciwników dostajemy punkty oraz losowo mogą wypaść nam bonusy. Zdobyte punkty można wydać w sklepie na zakup dodatkowych żyć, tarcz, poziomów ataku. Po zdobyciu określonych ilości punktów gra automatycznie zwiększa poziom trudności co prowadzi do zwiększania prędkości ruchów przeciwników, więc musimy walczyć z coraz silniejszymi przeciwnikami. Porażka w postaci utraty 3 żyć skutkuje utratą statku i musimy zacząć rozgrywkę od nowa.

Użyte oprogramowanie/technologie

Diagramy: Visual Paradigm

Język: Python

Środowisko: Visual Studio Code

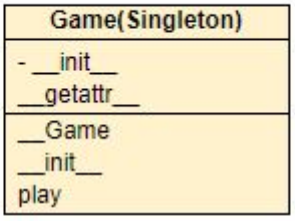


Użyte wzorce

1. KREACYJNE:
    a. Singleton
2. Strukturalne
    a. Flyweight(pyłek) +
    b. Decorator+
3. Czynnościowe
    a. Command +
    b. Strategy +
    c. State +

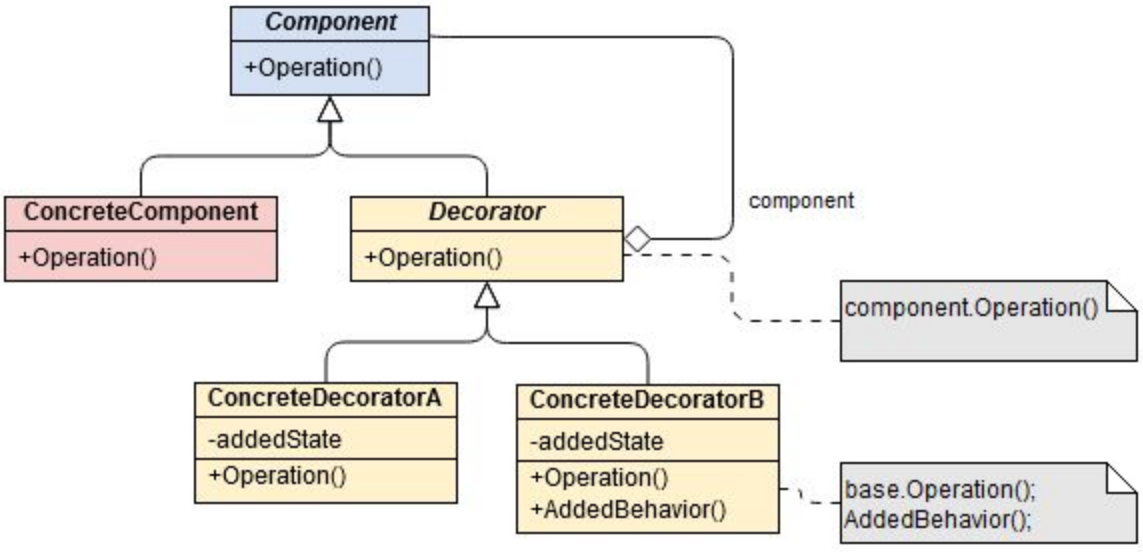
Wzorzec Singleton

Stworzenie Klasy Gry wymaga użycia wzorca singleton, ponieważ chcemy użyć w niej pole statyczne, prywatny konstruktor oraz pobieranie instancji. Użycie wzorca Singleton ogranicza możliwość tworzenia obiektów danej klasy do wybranej instancji oraz umożliwi nam on globalny dostęp do zawartych danych.



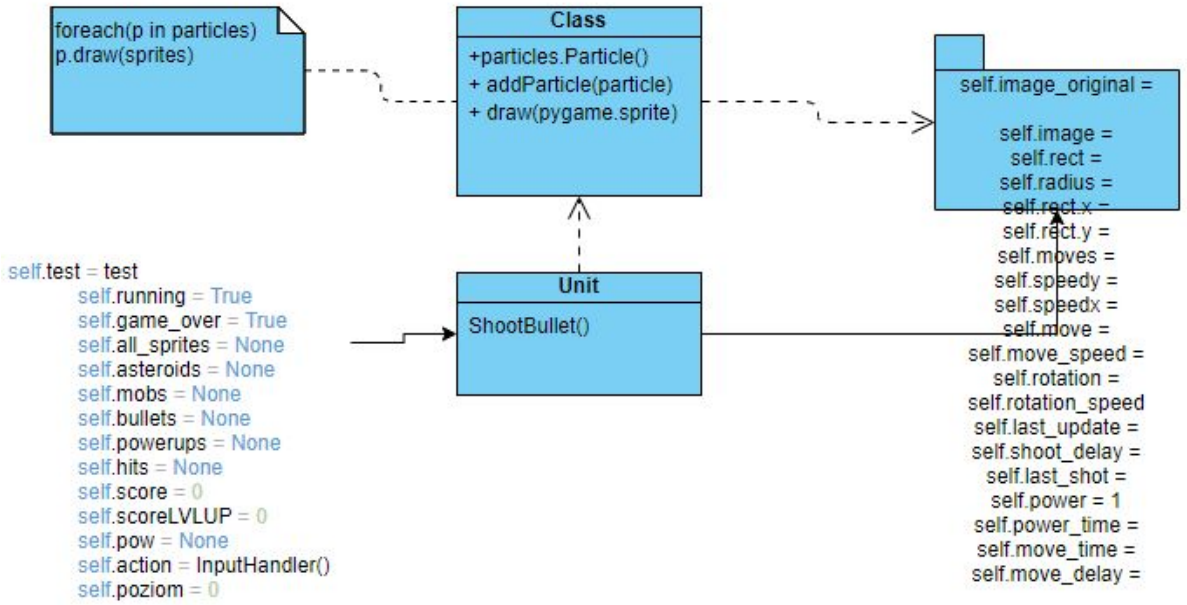
Wzorzec Decorator

Wzorzec decorator został przez nas użyty w postaci rozwinięcia rozgrywki, zdefiniowaliśmy go na zasadzie utworzenia menu zakupów, do którego możemy wejść za pomocą klawisza P w dowolnym momencie rozgrywki. Wzorzec ma za zadanie wprowadzić zmiany w statku gracza, takie jak: dodanie żyć gracza, zwiększenie poziomu tarczy, zmiana typu pocisków(pojedyńczy strzał -> podwójny strzał -> potrójny strzał -> 5-krotny strzał), podniesienie poziomu rozgrywki.



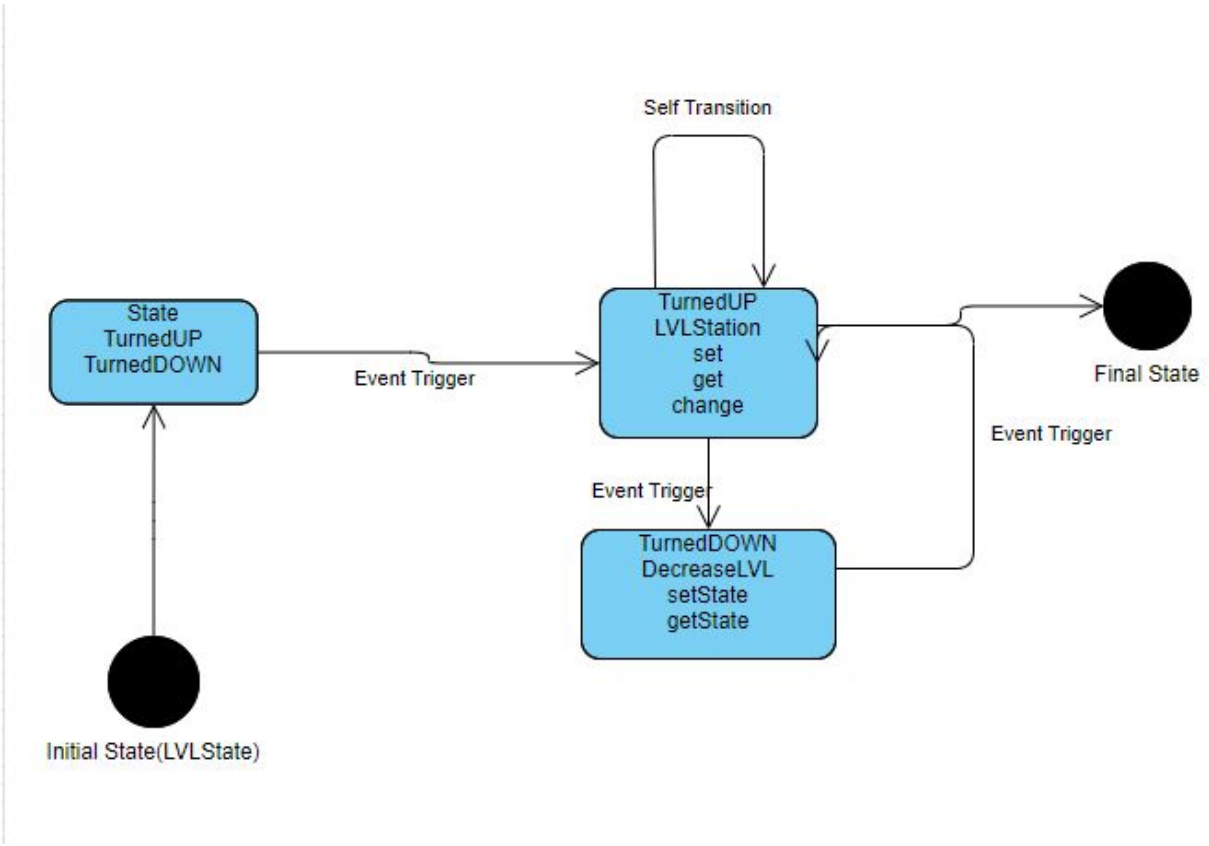
Wzorzec Flyweight użyty pośrednio

Pomimo braku bezpośredniej definicji pyłku w kodzie projektu, udało nam się wykorzystać pseudo kod pyłku w postaci korzystania z all_sprites. Czyli tworzenia obiektów w oparciu o gotowy szablon klasy wykorzystywanej. Stosując ten wzorzec jesteśmy w stanie tworzyć większą ilość obiektów o podobnych parametrach nie tracąc przy tym wydajności.



Wzorzec State

Wzorzec stan jest wykorzystywany przez nas do zwiększania poziomu trudności rozgrywki(jest nieskończenie wiele poziomów trudności). Korzystamy z niego przy wywołaniu komendy zakupu wyższego poziomu lub poprzez automatyczne zmiany poziomu trudności. Jest to stan obiektowy, zmieniający prędkość statków przeciwnika.



Wzorzec Command

Wzorzec command został przez nas wykorzystany bezpośrednio w klasie Player w postaci systemu kontroli kliknięć użytkownika. Umożliwia nam poruszanie oraz automatyczne zmiany stanu pojazdu gracza.

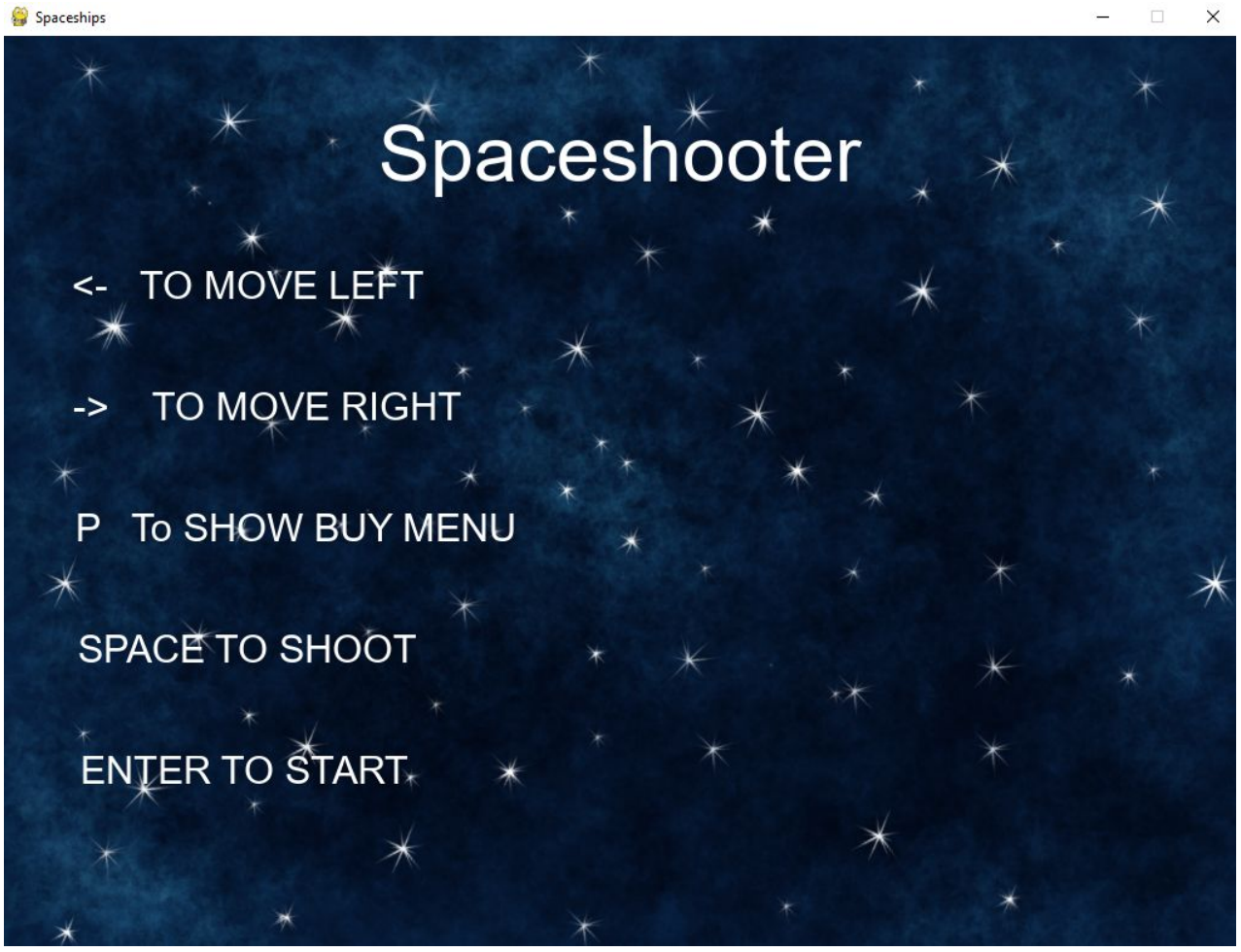
```
class Action(metaclass=ABCMeta):    # Command interface

    @staticmethod
    @abstractmethod
    def execute():
        pass

class MoveRightCommand(Action):     # ConcreteCommand

    def __init__(self, receiver):
        self.receiver = receiver

    def execute(self):
        self.receiver.move_right()

class MoveLeftCommand(Action):      # ConcreteCommand

    def __init__(self, receiver):
        self.receiver = receiver

    def execute(self):
        self.receiver.move_left()

class ShootCommand(Action):     # ConcreteCommand

    def __init__(self, receiver):
        self.receiver = receiver

    def execute(self):
        self.receiver.shoot()
```

Wzorzec Strategy

Korzystając z tego wzorca mamy zaimplementowane wzorce ruchowe przeciwników takie jak poruszanie w koło osi centralnej statku.

```python
class Strategy(metaclass=ABCMeta):

    @abstractmethod
    def move_strategy(self):
        pass
        print("Error: Strategy not chosen")

class XShiftStrategy(Strategy):
    def move_strategy(self):
        pass
        print("XShiftStrategy")

class StraightAtPlayerStrategy(Strategy):
    def move_strategy(self):
        pass
        print("StraightAtPlayerStrategy")

class RotationStrategy(Strategy):
    def move_strategy(self):
        pass
        print("RotationStrategy")

class LVL1Strategy(Strategy):
    def move_strategy(self):
        pass
        print("LVL1Strategy")

class LVL2Strategy(Strategy):
    def move_strategy(self):
        pass
        print("LVL2Strategy")

class LVL3Strategy(Strategy):
    def move_strategy(self):
        pass
        print("LVL3Strategy")
```

Menu Startu Gry



W oknie Menu_Start na samej górze znajduję się nazwa gry

Strzałka w lewo - Ruch w lewo

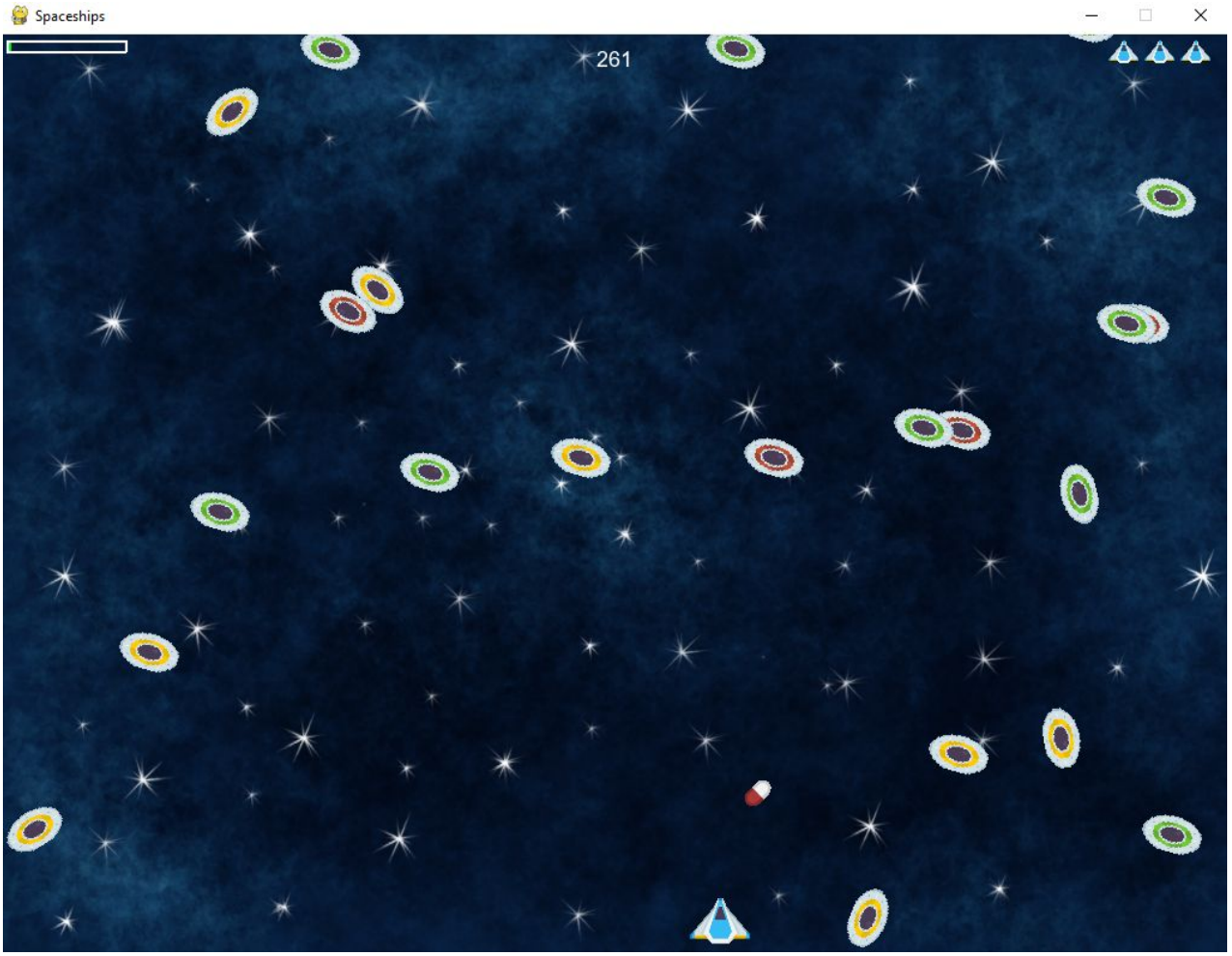
Strzałka w prawo - Ruch w prawo

P - Uruchomienie menu zakupów”SKLEP”

Spacebar - Strzał

Enter - Rozpoczęcie rozgrywki

Okno rozgrywki



Na dole ekranu znajduje się statek gracza

Ruchy przeciwników polegają na stałej sekwencji.
1. sekwencja ruchów obejmuje obracanie wokół własnej osi co daje możliwości wykorzystania falistych ruchów przypominających spłaszczoną spiralę

2. sekwencja stałego ruchu korzysta z wielu ograniczników dających złudzenie odbijania się statków od niewidocznych ścian urozmaicając stan rozgrywki

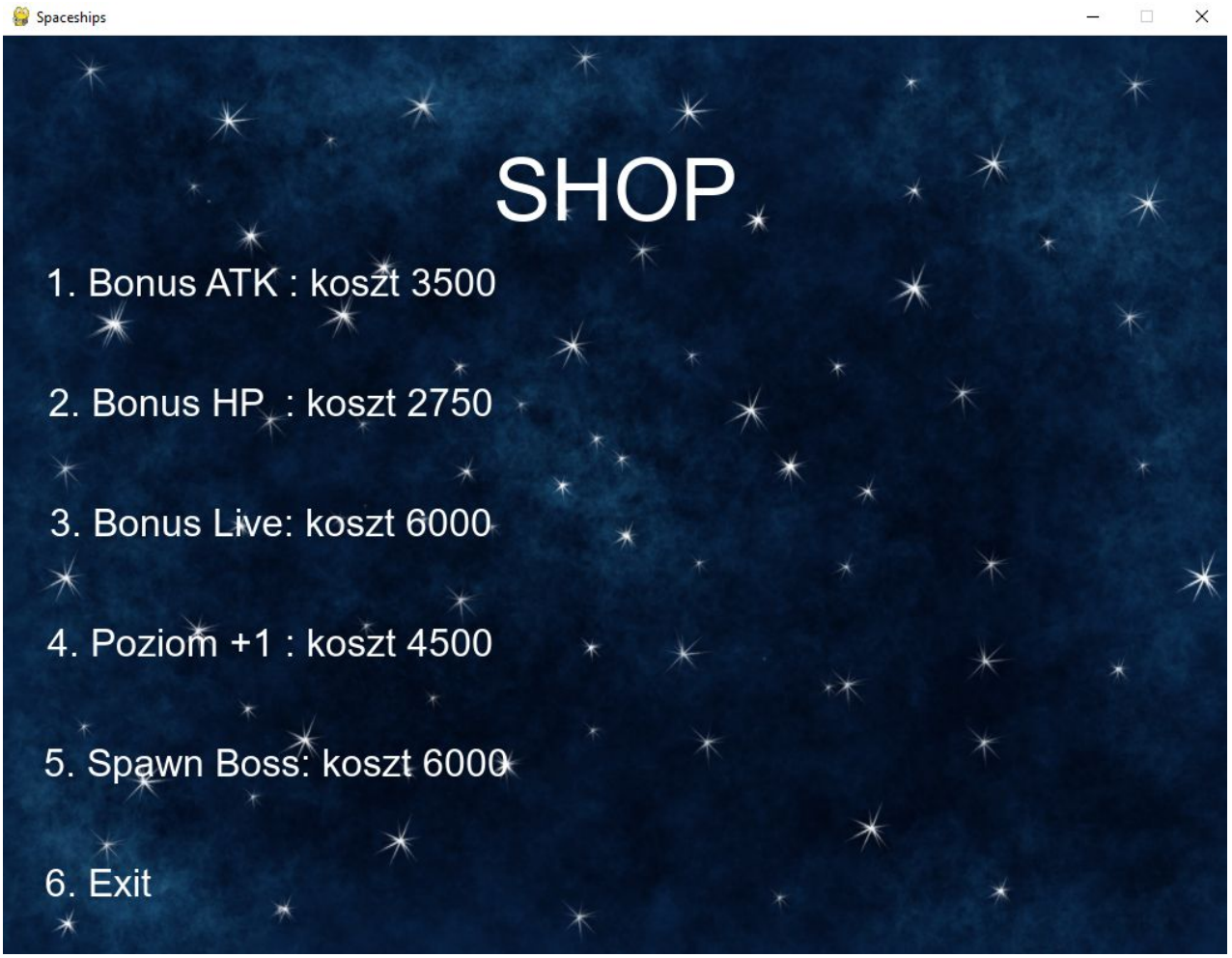
3. model prędkości umożliwia stałe zmiany prędkości statków przeciwników wraz z poziomami.

W lewym górnym rogu znajduje się aktualny poziom tarczy gracza

Na środku w górze ekranu znajduje się ilość posiadanych punktów

W prawym górnym rogu znajduje się posiadana ilość żyć gracza

Okno Sklepu



1. Bonus ATK: Naciśnięcie klawisza 1 spowoduje zwiększenie poziomu ataku o jeden poziom w górę odejmując koszt od ilości dotychczasowo zdobytych punktów
2. Bonus HP: Naciśnięcie klawisza 2 spowoduje zwiększenie aktualnego poziomu zdrowia do 100
3. Bonus Live: Naciśnięcie klawisza 3 spowoduje zwiększenie maksymalnej ilości żyć(max.3)
4. Poziom +1: Naciśnięcie klawisza 4 spowoduje zwiększenie poziomu trudności o 1

6. Naciśnięcie klawisza 6 lub dowolnego nieużytkowego przycisku spowoduje automatyczne wyjście z menu zakupów

Podział pracy

Michał Ziółkowski:

- **wzorce projektowe: Strategy,Flyweight, Decorator**
- **stworzenie menu zakupów**
- **Ogólny wygląd aplikacji**
- **Dostosowanie grafik wykorzystywanych przez projekt**
- **menu controller**
- **Klasa Player**
- **system obsługi Buffów**
- **Wykorzystanie pygame w celu aktywowania aplikacji na poziomie okienkowym**

Dawid Pawłowski:

- **wzorce projektowe: Command, State**
- **Move controller**
- **wprowadzenie wypisywania komend używanych przez wzorce**
- **Wprowadzenie logicznego podziału aplikacji**
- **Klasa Asteroid**
- **Obsługa all_sprites**
- **optymalizowanie płynności odświeżania**

Problemy w trakcie pisania pracy

Naszym głównym problemem na początku było napisanie projektu bez wykorzystania wzorców co skutkowało optycznie dobrze wyglądającym programem. Jednak podczas wprowadzania wzorców ujawniło się wiele błędów, z którym przeszliśmy ciężką drogę aby je poprawić lub zastąpić moduły wzorcami. Musieliśmy w naszym projekcie trochę edytować ogólnie przyjęte normy wzorców projektowych np wzorca state lub flyweight.