Projekt - gra 𝕆ld-𝕊chool 𝕊hooter

Programowanie obiektowe

Michał Górniak

7 czerwca 2018

1 Wstęp

𝕆ld-𝕊chool 𝕊hooter jest grą napisaną w języku Java jako projekt semestralny z programowania obiekotwego na zajęcia w Instytucie Informatyki UWr. Gra jest przeznaczona dla dwóch osób. W grze gracze toczą krótkie pojedynki w swoich statkach. Każdy pojedynek toczy się do trzech wygranych. Grafika gry jest dwuwymiarowa, a sama gra jest dość dynamiczna. Użytkownicy mają do wyboru dziewięć postaci.

2 Instrukcja obsługi

2.1 Sterowanie

𝕆ld-𝕊chool 𝕊hooter jest sterowany wyłącznie klawiaturą. Gracz pierwszy i drugi mają osobne przyciski funkcyjne.

Gracz 1:

1. sterowanie i poruszanie się po menu – **W S A D**
2. przycisk strzału – **lewy ALT**
3. przycisk ataku specjalnego – **lewy SHIFT**

Gracz 2:

1. sterowanie i poruszanie się po menu – **strzałki**
2. przycisk strzału – **SLASH**
3. przycisk ataku specjalnego – **. (PERIOD)**

2.2 Przebieg gry

Po uruchomieniu programu pojawia się okienko powitalne gry. Wprost z niego gracze przechodzą do okienka wyboru tła pojedynku. Dostępne są cztery możliwe grafiki. Po wybraniu planszy gracze przechodzą do panelu wyboru postaci. Tutaj każdy gracz operuje niezależnie na swojej połówce. Użytkownik przegląda postacie i sprawdza ich statystyki. Gdy pierwszy z graczy dokona wyboru jego panel przyciemnia się i zaznacza, że ten gracz jest już gotowy do pojedynku. Rozgrywka zaczyna się, gdy drugi gracz podejmie decyzję.

Właściwa gra polega na wygraniu jako pierwszy trzech rund. W każdej rundzie gracze zaczynają w losowym miejscu na swojej połowie planszy. Podczas rozgrywki gracze mogą dowolnie poruszać się po całej planszy i strzelać. Raz na jakiś czas gracz dostaje supermoc. Po aktywacji przyciskiem funkcyjnym wszystkie pociski wystrzelone przez gracza w przeciągu kilku sekund od aktywacji są przyspieszone i zadają podwójne obrażenia. Gracze podczas pojedynku mogą także łapać losowo spadające serca, aby zregenerować swoje zdrowie. Nie mogą jednak uzyskać więcej zdrowia, niż mieli na początku rundy. Punkt za rundę zdobywa gracz, który zabierze jako pierwszy całe życie drugiego gracza. W przypadku remisu obaj gracze zdobywają punkt.

3 Analiza obiektowa

3.1 Lista klas z opisem

3.1.1 Frame

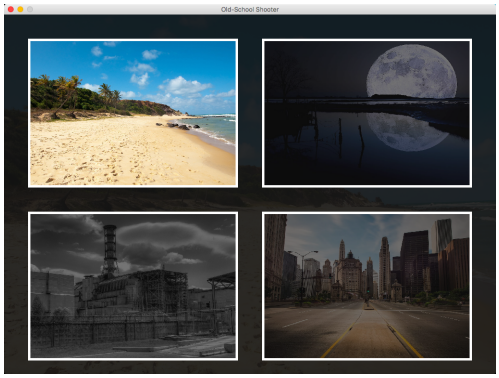
Klasa dziedzicząca po *JFrame*. Ustawia domyślne wartości dla okna (resizable(false), wymiary, wyjście).

3.1.2 BackgroundFrame

Okno wyboru tła. Dziedziczy po *Frame*. Ma w sobie tylko jeden panel *BackgroundPanel*.

3.1.3 BackgroundPanel

Panel wybrou tła. Na ekranie pojawiają się cztery prostokąty z pomniejszonymi grafikami do wyboru. Wybieramy tło poruszając się klawiszami ruchu i zatwierdzamy przyciskiem funkcyjnym. Aktywne w danym momencie tło jest podświetlone, a jego przyciemniona wersja jest widoczna w tle.



Rysunek 1: Panel wyboru tła.

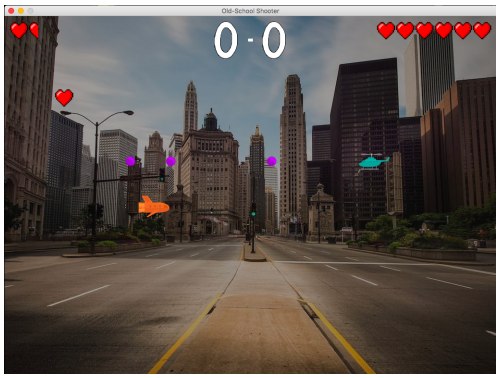
3.1.4 BoardFrame

Okno rozgrywki (planszy). Dziedziczy po *Frame*. Ma w sobie tylko jeden panel *BoardPanel*.

3.1.5 BoardPanel

Panel rozgrywki. Zaimplementowanie przy użyciu głownie rysowania planszy, wstawiania obrazków i ustawiania różnych timerów. W rozgrywce występują też czynniki losowe. Spadające

serca pojawiają się w losowym miejscu, gracze startują na losowych pozycjach na swoich połowach. Co określoną stałą plansza odświeża się, przerysowując wszystko od nowa, co tworzy efekt animacji. W klasie tej trzymani są bohaterowie. W górnej części panelu mieszczą się stany zdrowia bohaterów oraz aktualny wynik. Przed każdą rundą pojawia się animacja odliczania do rozpoczęcia rundy. Po zakończeniu ostatniego pojedynku pojawia się animacja *Game Over*.



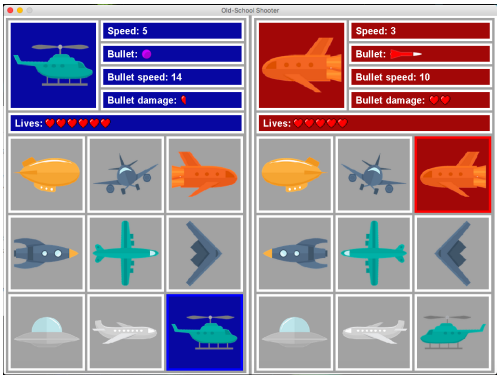
Rysunek 2: Panel pojedynku.

3.1.6 CharacterFrame

Okno wyboru postaci. Dziedziczy po *Frame*. Ma w sobie tylko jeden panel *CharacterPanel*.

3.1.7 CharacterPanel

Panel wyboru postaci. Są to dwie kopie (na odpowiednich połowach) menu wyboru postaci (dla gracza lewego i prawego). W górnej części pojawia się aktualnie zaznaczona postać i jej statystyki. W dolnej części jest panel wyboru bohatera. Gracze w tej części wybierają niezależnie. Gdy jeden z graczy dokona wyboru jego okienko przyciemnia się i wyświetlany jest komunikat *Ready*. Menu zostało to zaimplementowane jako grafiki i rysowanie prostych obiektów, a przyski są tak naprawdę przemalowywane za każdym razem, gdy któryś z graczy wciśnie odpowiedni klawisz. Dla uatwienia implementacji w klasie jest funkcja rysująca połowę ekranu (z odpowiednimi parametrami).



Rysunek 3: Panel wyboru postaci.

3.1.8 GameOverFrame

Okno uruchamiane po zakończeniu pojedynku. Dziedziczy po *Frame*. Ma w sobie tylko jeden panel *GameOverPanel*.

3.1.9 GameOverPanel

Prosty panel uruchamiany po zakończeniu gry. Jest na nim popularna grafika *Game Over* oraz wynik pojedynku i miniaturki graczy. Dostępne są dwa przyciski: jeden do powrotu do menu głównego, a drugi do zakończenia gry.

Rysunek 4: Panel *Game Over*.

3.1.10 MenuFrame

Okno uruchamiane przy starcie programu. Dziedziczy po *Frame*. Ma w sobie tylko jeden panel *MenuPanel*.

3.1.11 MenuPanel

Panel startowy. Za każdym uruchomieniem pojawia się losowe tło. Co kilka sekund zmieniają się miniaturki po lewej i prawej stronie oraz ich pozycje. Zmienia się także kolor napisu na przycisku *New game*.

Rysunek 5: Panel menu.

3.1.12 MovingCharacter

Klasa zawierająca podstawowe wartości dla poruszającego się po planszy obiektu (np. koordynaty, obrazek).

3.1.13 Hero

Klasa bohatera. Dziedziczy po *MovingCharacter*. Zawiera metody poruszania się bohatera oraz trzyma jego aktualne zdrowie, szybkość, a także listę wystrzelonych przez niego pocisków.

3.1.14 Bullet

Klasa pocisku. Dziedziczy po *MovingCharacter*. Zawiera prędkość posicku.

3.1.15 Superbullet

Podklasa pocisku. Pociski te zadają podwójne obrażania oraz są dodatkowo przyspieszone.

3.1.16 FallingLife

Klasa implementująca losowo spadające serca podczas rozgrywki.

3.1.17 GameConst

Klasa z ustawieniami gry. Jest to klasa z prywatnym konstruktorem i publicznymi finalnymi statycznymi zmiennymi.

3.1.18 HeroStats

Klasa ze statystykami bohaterów oraz ich pocisków.

3.1.19 MyImage

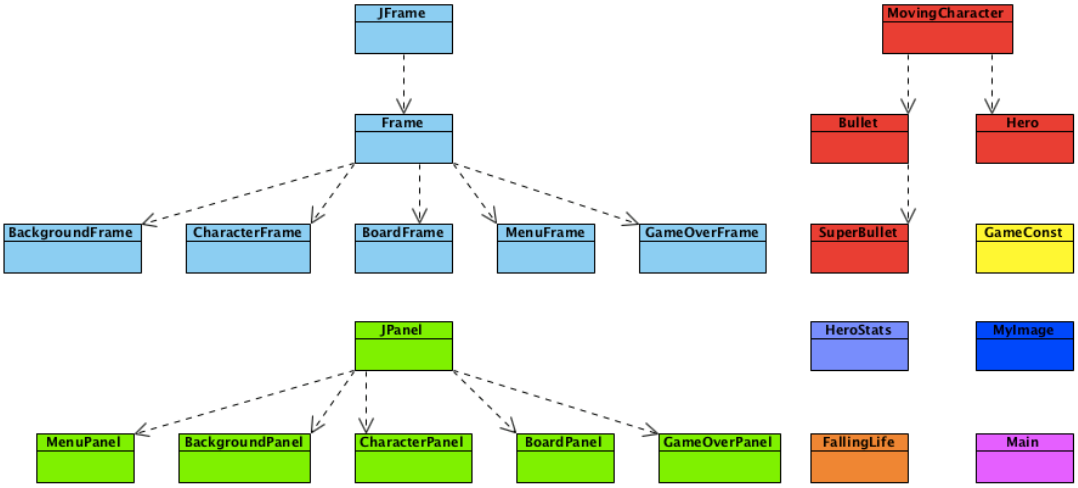
Pomocnicza klasa do obsługi obrazów. Trzyma obiekt klasy Image oraz pola i metody potrzebne do wygodnej obsługi obrazu.

3.1.20 Main

Klasa uruchamiająca program.

3.2 Diagram klas

Poniżej przedstawiam diagram klas projektu:



Rysunek 6: Diagram klas.

3.3 Realizowana funkcjonalność

3.3.1 Użytkownik

Każdy użytkownik ma taki sam dostęp do aplikacji. W grze nie zostały umieszczone żadne ukryte umiejętności, które faworyzują któregoś z graczy. Rozgrywka jest symetryczna dla obu graczy. Każdy użytkownik ma zatem dostęp do wszystkich funkcji programu.

3.3.2 Programista

Aplikacja jest przystosowana do dalszego rozwoju. Więcej o tym w rozdziale **Dalsza rozbudowa**. Poza rozszerzeniem kodu użytkownik może dowolnie zmieniać grafiki postaci i tła podmieniając odpowiednie pliki w folderze *images*. Poza tym programista może modyfikować klasy *GameConst* i *HeroStats*, aby dopasować statystki postaci i ustawienia gry do własnych upodobań i testów.

4 Dalsza rozbudowa

Gra była pisana w dość krótkim czasie, istnieje zatem wiele możliwości rozwoju. Programista ma w tym przypadku pozostawioną dużą dowolność. Modyfikacje można rozpocząć od zmiany statystyk i ustawień gry. Nietrudno też dodać nowe strzały i ataki specjalne. Dla przykładu strzały można rozbudowywać tworząc podklasy klasy *Bullet*, dodając im nowe grafiki i zmieniając parametry lub sposób działania. Grę można też rozszerzyć o dodatkowe postacie, a postaciom tym można dodać nowe umiejętności.