Space Invaders

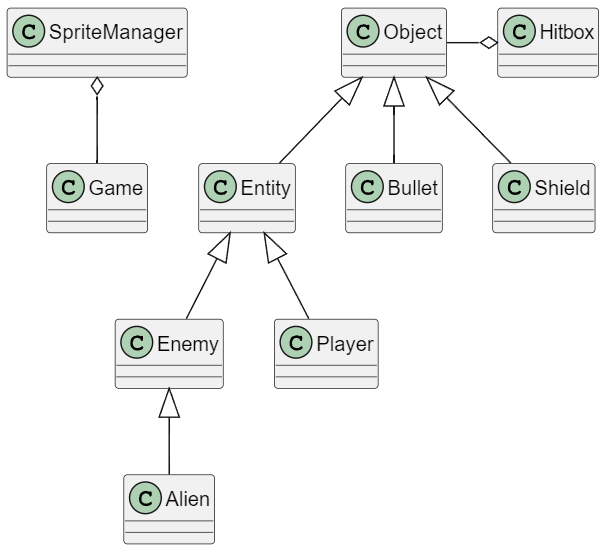
Autor: Sebastian Kubik

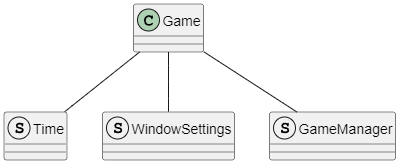
1. **Opis gry:**

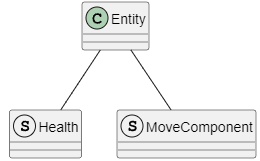
Jest to prosty „klon” gry Space Invaders

<https://pl.wikipedia.org/wiki/Space_Invaders>

1. **Wymagania dla systemu:**
   1. **Funkcjonalne:**
2. Gra:
   1. Wyświetlanie okna gry.
   2. Wyświetlanie przeciwników na ekranie.
   3. Wyświetlanie gracza na ekranie.
   4. Wyświetlenie tarcz na ekranie.
   5. Wyświetlanie pocisków na ekranie.
   6. Wyświetlanie ilości zdobytych punktów na ekranie.
   7. Wyświetlanie ilości punktów zdrowia gracza na ekranie.
3. Gracz:
   1. Możliwość sterowania statkiem prawo/ lewo.
   2. Możliwość strzelania.
   3. Możliwość trafienia przeciwnika.
   4. Możliwość trafienia tarczy.
   5. Możliwość uzyskania punktu po pokonaniu przeciwnika.
4. Przeciwnik:
   1. Możliwość poruszania się w lewo lub prawo.
   2. Możliwość przesunięcia się w dół, gdy dotrze do krawędzi ekranu.
   3. Możliwość oddania strzału( losowa ).
   4. Zresetowanie po dotarciu do linii tarcz.
   5. Możliwość przyśpieszenia po pokonaniu przeciwnika.
   6. Zresetowanie przeciwników po pokonaniu ostatniego przeciwnika na planszy.
5. Tarcza
   1. Redukcja pancerza po otrzymaniu obrażeń.
   2. Zresetowanie po dotarciu przeciwnika do linii tarcz.
   3. Wyświetlanie ilości pancerza.
   4. **Niefunkcjonalne:**
6. Kod:
   1. Kod powinien być czytelny.
   2. Powinna być możliwość dodania nowych funkcjonalności.
7. Rozgrywka:
   1. Gra powinna być łatwa w sterowaniu.
   2. Gracz nie powinien mieć problemu zrozumieniem zasad gry.
8. Wydajność:
   1. Gra powinna działać na Windows 10 w wersji 32 i 64 bitowej.
   2. Gra powinna być zablokowana na 120 klatek na sekundę.
   3. Gra powinna chodzić płynie i szybko.
9. **Schematy UML:**







1. **Opis Programu:**

Program korzysta z następujących zmiennych instancyjnych z biblioteki SFML:

Sf::RenderWindow, sf::RenderTarget – służą do narysownia okna oraz wszystkich elementów w tym oknie.

Sf::RectangleShape – jest to obiekt o kształcie prostokąta.

Sf::Color – pozwala nam zmienić kolor np. RectangleShape.

Sf::Sprite oraz sf::Texture – służą do wczytania obrazka.

Klasa Game jest rdzeniem aplikacji to tutaj są ustawiane parametry okna oraz są wywoływane odpowiednie konstruktory. W metodzie run wywołujemy metody Update i Render. Update służy do zaktualizowania wszystkich obiektów w programie. W tej metodzie wywołujemy odpowiednie metody Update poszczególnych obiektów. W metodzie Render skupiamy się na wywołaniu metod do rysowania obiektów na ekranie.

Klasa SpriteManager służy do przechowywania spritów. Wywołując metode LoadSprite wczytujemy nowe sprite, a za pomocą metody DrawSprite rysujemy dany obiekt na ekranie.

Metoda DrawSprite jako jeden z parametrów bierze informacje o położeniu sprite na ekranie i na podstawie tych informacji rysuje sprite w odpowiednim miejscu.

Klasa Object jest klasą z której dziedziczą wszystkie obiekty. Klasa ta posiada zmienną instancyjną Hitbox który określa hitbox danego obiektu.

Klasy Bullet, Shield oraz Entity bezpośrednio dziedziczą od objekt.

Klasa Enemy i Player dziedziczą po Entity.

Po klasie Enemy dziedzycza Alien, ale jest również możliwość dodania nowych przeciwników którzy tak samo jak Alien dziedziczą po Enemy. Dzięki temu w klasie Game mamy zmienną vector która przechowuje zmienne typu enemy i na tych zmiennych wywoływane są odpowiednie metody takie jak Update lub Render.

1. **Sterowanie:**

A – ruch w lewo.

D – ruch w praw.

Space – strzał.